ESCAPE STRATEGIES ALGORITHM (ESSA)
UN NUEVO ALGORITMO META HEURÍSTICO DE OPTIMIZACIÓN GLOBAL PARA PROBLEMAS DE VARIABLE REAL, INSPIRADO EN LA INTERACCIÓN DEPREDADOR-PRESA.

MARIO ALBERTO MENDOZA VILLALBA

UNIVERSIDAD DE CÓRDOBA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA INDUSTRIAL
MONTERÍA, CÓRDOBA
2016.

ESCAPE STRATEGIES ALGORITHM (ESSA)
UN NUEVO ALGORITMO META HEURÍSTICO DE OPTIMIZACIÓN GLOBAL PARA PROBLEMAS DE VARIABLE REAL, INSPIRADO EN LA INTERACCIÓN DEPREDADOR-PRESA.

MARIO ALBERTO MENDOZA VILLALBA

Trabajo de grado presentado, en la modalidad de Trabajo de Investigación y/o Extensión, como parte de los requisitos para optar al Título de Ingeniero Industrial

Director:
Ing. JORGE MARIO LÓPEZ PEREIRA, M.Sc.

UNIVERSIDAD DE CÓRDOBA
FACULTAD DE INGENIERÍAS
PROGRAMA DE INGENIERÍA INDUSTRIAL
MONTERÍA, CÓRDOBA
2016.

La responsabilidad ética, legal y científica de las ideas, conceptos y resultados del proyecto, serán responsabilidad de los autores.

Artículo 61, acuerdo N° 093 del 26 de noviembre de 2002 del consejo superior.

Nota de aceptación

_____

_____

_____

_____

_____

Firma del jurado

_____

Firma del jurado

A Dios Todopoderoso
A mis padres Rito Mendoza Narváez y
Mary Villalba Soto

A Dios todopoderoso, por cuya bondad obtenemos la gracia de vivir y ver nuestros sueños hacerse realidad. A mis padres, que son un regalo invaluable del altísimo, a quienes debo todo.

Agradecimiento especial a: Ing. Jorge Mario López Pereira, por cuyo ejemplo he crecido académica y personalmente.

Agradecimientos:

Al Dios de bondad que hace posibles todas las cosas

A mis amados padres, Regalo maravilloso de Dios en mi vida.

A los licenciados Alba Alean , Rairo Wilchez, Álvaro Díaz., QPD. Por forjar los cimientos de mí formación académica.

A todos aquellos maestros que en el trascurso de varios años de estudios universitarios me guiaron con sus conocimientos y ejemplo personal.

A todos aquellos familiares y amigos que de alguna manera ayudaron a hacer esto posible.

TABLA DE CONTENIDO.

Pág.

LISTADO DE TABLAS

Pág.

Pág.

LISTADO DE GRÁFICOS

Pág.

RESUMEN

El reto de mejorar las técnicas de optimización sigue siendo, hoy más que nunca, un tema que reviste gran relevancia en múltiples campos de la ciencia y la ingeniería, pues, cada día surgen más problemas que requieren el uso de diversos métodos de optimización; además, la complejidad de los modelos aumenta a medida que los sistemas se complejizan, obligando esto a que las técnicas de solución tengan que ser cada vez más eficientes; es por ello que el desarrollo de nuevos métodos aproximados (heurísticas y meta heurísticas) ha proporcionado una importante alternativa en la solución de problemas de gran complejidad por su versatilidad, aplicabilidad y eficiencia. Asimismo, el presente trabajo propone un nuevo algoritmo meta heurístico de optimización global llamado ESCAPE STRATEGIES ALGORITHM (ESSA), inspirado en la interacción entre depredador y presa, y en como las presas tratan de evadir al depredador; el cual se prueba en funciones reales, comparándolo con meta heurísticas de codificación real ampliamente referenciadas en la literatura en la literatura, obteniendo (ESSA) una calidad de respuesta igual o mejor en sus soluciones para un tiempo estándar en múltiples instancias de prueba, además de igualar en puntaje típico estandarizado a los mejores algoritmos de comparación y superando a varios de los más citados. Adicionalmente, el análisis descriptivo evidencia que, en efecto, ESSA es el algoritmo que resultó vencedor en el mayor número de instancias de prueba.

Palabras Clave: Optimización, heurística, meta heurística.

ABSTRACT

The challenge of improving optimization techniques remains, today more than ever, a subject of great importance in many fields of science and engineering. Because, every day more problems arising that require the use of various optimization methods, in addition, the complexity of the methods increases as systems become more complex, forcing it to solution techniques have to be increasingly efficient; that is way the development of new approximate methods (heuristics and metaheuristics) has becoming an important alternative in solving complex problems for its versatility, applicability and efficiency. Also, this paper proposes a new global optimization metaheuristic algorithm named ESCAPE STRATEGIES ALGORITHM (ESSA), inspired by the interaction between predator and prey, and how the preys try to evade to predator; which is tested in real functions, comparing with real encoding metaheuristics widely referenced in literature, getting (ESSA) a equal to or better performance in their solutions to a standard time on multiple test instances, besides equaling in typical standardized score to the best comparison algorithms and overcoming for several of the most cited. In addition, the descriptive annalyzis shows, in fact, ESSA is the algorithm that wins in biggest quantity of test instances.

Key words: Optimization, heuristic, metaheuristic.

1. INTRODUCCIÓN

En la actualidad, la limitación de los recursos, el aumento de la población y de las necesidades ha obligado a las sociedades a convertir en una prioridad la eficiencia en el manejo de los recursos, dándole a la disciplina de la optimización un carácter imprescindible en la consecución de dicha eficiencia. (Lobo y Castro 2011) Sin embargo, el surgimiento de nuevos problemas de optimización y el aumento en la complejidad de los mismos fomentan la creciente necesidad de diseñar nuevos y mejores métodos de optimización que permitan alcanzar soluciones de alta calidad con un menor consumo de tiempo computacional; es por ello que las técnicas meta heurísticas han cobrado fuerza en las últimas décadas, aplicándose a una amplia gama de problemas de optimización.

En vista de lo anterior, el presente trabajo propone un nuevo método meta heurístico de optimización global para problemas de codificación real, llamado ESCAPE STRATEGIES ALGORITHM (Estrategias de Escape), el cual simula la interacción depredador-presa aprovechando la "inteligencia de enjambre". En este trabajo se describe el diseño del algoritmo, pensado para alcanzar mejor calidad en las soluciones en el menor tiempo posible.

Se describe además el experimento de comparación, en el cual se contrastó el nuevo método versus 4 algoritmos ampliamente referenciados en la literatura (Enjambre de partículas PSO, Algoritmo genético GA, Evolución diferencial DE y Recocido simulado

SA), se comparó también con la meta heurística Eidético eWSA, una evolución del algoritmo Búsqueda de lobo WSA, por ser uno de los métodos más recientes y mostrar similitud con ESSA en cuanto a su inspiración, además, se incluyó en la comparación un algoritmo de generación aleatoria RANDOM. La comparación se da a través de métodos de inferencia estadística utilizando 20 funciones de prueba en 53 instancias.

2. PLANTEAMIENTO DEL PROBLEMA

2.1.DESCRIPCIÓN DEL PROBLEMA

En los últimos años, los algoritmos meta heurísticos han sido considerados poderosos métodos en la solución de problemas de optimización (Bidar y Rashidy Kanan 2013), muchos problemas de ingeniería representan cierta dificultad para su solución, y por su complejidad no pueden ser resueltos a través de los métodos tradicionales (exactos), por lo que ha sido necesario el diseño de meta heurísticas para darles solución (Yazdani y Jolai 2015) , no obstante, el problema de aportar mejores soluciones, haciendo uso de un menor tiempo de computo a través del diseño de nuevas meta heurísticas sigue siendo un problema abierto (Hajipour et al. 2012). Es por ello que el presente estudio busca la formulación de un nuevo algoritmo meta heurístico de optimización global que aporte soluciones de alta calidad para problemas de variable real en un tiempo de cómputo racional.

2.2.PREGUNTAS DE INVESTIGACIÓN

- ¿Cuál es el nivel de calidad de respuesta que proporciona el nuevo algoritmo meta heurístico propuesto?
- ¿Representan esta calidad de respuesta alguna mejoría en comparación con las metas heurísticas más referenciadas en la literatura?
- ¿Cuáles son los valores de los parámetros que optimizan el funcionamiento del algoritmo propuesto?

3. JUSTIFICACIÓN

A nivel mundial, en las últimas décadas, el desarrollo de nuevos algoritmos de búsqueda aplicados a los problemas de optimización ha experimentado un rápido crecimiento, a pesar de ello, dichos métodos siguen aún rezagados con respecto a ciertos factores como la calidad de sus respuestas y el costo en tiempo computacional, en detrimento de la deseada eficiencia en el proceso de obtención de las soluciones (Lobo y Castro 2011). De allí la gran importancia que para la ciencia de la optimización posee el desarrollo continuo de nuevas técnicas y algoritmos que aporten soluciones de alta calidad y un bajo costo en tiempo computacional.

Las posibilidades de aprovechar las ventajas que los algoritmos y los nuevos métodos de optimización aportan es prácticamente ilimitada, siendo utilizables en infinidad de aplicaciones científicas, industriales, económicas, tecnológicas, etc. Dando esto un carácter diverso y ampliamente adaptable. Además, en vista de que los problemas de optimización surgen en muchos campos diferentes, numerosos estudios se centran sobre todo en el desarrollo de nuevos métodos de optimización (Kuo y Zulvia 2015), lo que demuestra la gran aplicabilidad de estos métodos en casi cualquier rama de la ciencia, contribuyendo a resolver una amplia gama de problemas prácticos que ocupan a diversas organizaciones.

Además, los métodos aproximados (heurísticos y meta heurísticos) no son herramientas desconocidas para las aplicaciones científicas y organizacionales, y su uso está ampliamente difundido, de modo que la aplicabilidad de dichas técnicas está más que garantizada. Así lo sustenta (Salimi 2015): Los algoritmos meta heurísticos (MH) son términos bien conocidos en muchos campos de la ciencia. Específicamente cuando los métodos matemáticos comunes son incapaces de proporcionar una buena solución o la búsqueda por métodos exactos requiere una cantidad no razonable de tiempo (problemas NP, NP Hard). Hoy en día, muchos métodos MH están siendo propuestos y desarrollados Como podemos ver en (Hajipour et al. 2012): Estos algoritmos metaheurísticas han sido utilizados en una amplia gama de problemas de optimización como reconocimiento de patrón, aprendizaje de máquinas, la toma de decisiones en planificación industrial etc. pero no hay un algoritmo concreto para conseguir la mejor solución para todo" además (Hajipour et al. 2012) especifica: "BUSCAR NUEVOS ALGORITMOS META HEURISTICOS ES UN PROBLEMA ABIERTO".

En conclusión, el desarrollo de nuevas meta heurísticas de optimización sigue siendo muy necesario en la actualidad, en que las organizaciones, así como la ciencia y la tecnología afrontan retos cada vez más exigentes, a los cuales, los algoritmos aproximados proporcionan una excelente alternativa que goza de gran eficiencia, aplicabilidad y un amplio campo de acción.

4. OBJETIVOS

4.1.OBJETIVO GENERAL

- Diseñar un nuevo algoritmo meta heurístico de optimización global que en promedio presente una calidad de respuesta no inferior a la aportada por los algoritmos más utilizados en la solución de problemas de variable real para un tiempo de cómputo estándar.

4.2.OBJETIVOS ESPECÍFICOS

- Realizar la respectiva revisión de la literatura disponible, referente al diseño de algoritmos meta heurísticos y su correspondiente validación.
- Determinar cuáles son los algoritmos, funciones, parámetros e indicadores de comparación a utilizar en la respectiva validación del algoritmo propuesto.
- Diseñar la lógica del nuevo algoritmo meta heurístico e implementarla en el software MATLAB
- Realizar el correspondiente ajuste de parámetros que permitan al algoritmo propuesto alcanzar mejores soluciones.

- Determinar si existe diferencia estadística entre la calidad de las soluciones aportadas por el nuevo algoritmo propuesto y las metaheurísticas más referenciadas.

5. REVISIÓN DE LITERATURA

En el presente numeral se abordarán las temáticas referentes a la optimización, su definición y la clasificación de sus respectivos métodos, además se abordarán específicamente las técnicas de optimización aproximada (heurísticas y meta heurísticas), profundizando en el desarrollo de meta heurísticas a través de los años.

5.1.OPTIMIZACIÓN

Podemos entender la optimización como el proceso de encontrar la mejor solución posible a un problema determinado (Lobo y Castro 2011), no obstante, una definición más formal la aporta (Martí n.d.), que precisa: "encontrar el valor de unas variables de decisión para los que una determinada función objetivo alcanza su valor mínimo o máximo".

Si queremos abarcar un concepto más amplio, podemos acudir a (Lence 2007), quien define un problema de optimización partiendo de un conjunto de variables independientes, una serie de restricciones que delimitan valores aceptables de las variables, y una función objetivo que depende de estas variables.

5.2.Aplicabilidad y utilidad de la optimización.

El avance en la ciencia y la tecnología, así como los procesos de globalización traen consigo una gran cantidad de nuevos retos para las organizaciones, los cuales se suman a una amplia gama de problemas clásicos que atraen la atención de los interesados en diseñar y mantener sistemas eficientes y competitivos, es por ello que las técnicas de

optimización desempeñan un papel clave en el alcance de dichos objetivos, como lo evidencia (Salimi 2015), al afirmar que necesitamos optimización para minimizar el tiempo, el costo y el riesgo; para maximizar el lucro, la calidad o la eficiencia. (Salimi 2015) También señala que numerosas y complejas situaciones de la vida real requieren obtener mejores soluciones a los problemas que han surgido en muchos campos de la ciencia, como es el caso de la ingeniería, la economía y los negocios.

De esta manera, se hace evidente la importancia y vigencia de la optimización, en la búsqueda por asegurar una mayor eficiencia en los procesos y encontrar solución a una gran variedad de problemas en diversidad de ciencias.

5.3.Métodos de optimización:

En el transcurso de las últimas décadas se han desarrollado gran cantidad de métodos pro pendientes a dar solución a los problemas de optimización que se presentan a las organizaciones, dichos métodos, según (Kuo y Zulvia 2015), podrían ser clasificados en dos grandes grupos: Exactos y aproximados, los primeros aseguran el óptimo, los segundos se acercan al óptimo. Los métodos exactos requieren una mayor complejidad de cálculo y a menudo se dificulta resolver problemas complejos. Cuando la complejidad aumenta los métodos aproximados son más apropiados que los métodos exactos.

5.2. HEURÍSTICAS Y META HEURÍSTICAS:

En este numeral profundizaremos en los métodos aproximados de optimización, tanto heurísticos como meta heurísticos, abordando diversas definiciones y evidenciando su aplicabilidad.

5.2.1. Definición “Heurística”

En Inteligencia Artificial (IA) se emplea el calificativo heurístico, en un sentido muy genérico, para aplicarlo a todos aquellos aspectos que tienen que ver con el empleo de conocimiento en la realización dinámica de tareas (Santana et al. 2004). Sin embargo, para hacernos de un concepto de fácil comprensión recurramos a la definición propuesta por (Zanakis y Evans 1981): procedimiento simple, guiado por el sentido común, que se supone aportará buenas soluciones problemas difíciles que no necesariamente serán las óptimas, de manera fácil y rápida. Además, se usa el terminó heurístico para referirse a un procedimiento que trata de aportar soluciones a un problema con un buen rendimiento, en lo referente a la calidad de las soluciones y a los recursos empleados (Melián et al. 2003), (Pérez n.d.) nos da otra definición y además ilustra el origen del término "Heurística": "La idea más genérica del término heurístico está relacionada con la tarea de resolver inteligentemente problemas reales usando conocimiento. El término heurística proviene de una palabra griega con un significado relacionado con el concepto de encontrar y se vincula a la supuesta exclamación eureka de Arquímedes al descubrir su famoso principio". De esta manera, podemos entender las heurísticas como el punto de partida de las meta heurísticas, pues, según (García Sánchez n.d.) se utilizan cuando no existe un método exacto de solución, cuando aun existiendo un método exacto este requiere de mucho tiempo para alcanzar la solución óptima, cuando el tiempo es un factor limitante o como un paso intermedio para llegar a una solución inicial antes de aplicar otra técnica.

5.2.2. Definición "Meta heurística"

Según (Meng et al. 2015) podemos definir los algoritmos meta heurísticos como algoritmos estocásticos con componente de aleatoriedad (Exploración) y búsqueda local

(explotación). Sin embargo, se puede consultar a (Glover 1986) para una definición formal de meta heurística, que, es de anotar, que con dicho concepto se introdujo el término "meta heurística" (Santana et al. 2004). Santana, además, nos aclara la etimología del término meta heurística: se obtiene de anteponer a heurística el sufijo "meta" que significa "más allá" o "a un nivel superior". (Blum y Roli 2003) Aporta además, una definición bastante amplia, señalando que las metaheurísticas son estrategias de un nivel avanzado, las cuales permiten tener un mejor desempeño que las heurísticas tradicionales, generando nuevas soluciones iníciales para la búsqueda local de forma más inteligente que dando soluciones iníciales al azar; usando muchas veces el componente probabilístico para tomar decisiones. Inclusive, (Santana et al. 2004) afirma que la principal diferencia con la búsqueda al azar es que la aleatoriedad en los algoritmos metaheurísticas es usada de forma inteligente. Además (Meng et al. 2015) señala que con estos métodos se obtienen soluciones aceptables en un tiempo razonable, superando los problemas de no convergencia y no diferenciabilidad, por ello, los algoritmos meta heurísticos han atraído gran interés como alternativa de solución a los métodos tradicionales de optimización. (Sadic M. Sait n.d.) Aporta un listado de características generales de los algoritmos meta heurísticos:

- Son ciegas, no saben si llegan a la solución óptima. Por lo tanto, se les debe indicar cuándo deben detenerse. Son algoritmos aproximativos y, por lo tanto, no garantizan la obtención de la solución óptima.
- Aceptan ocasionalmente malos movimientos (es decir, se trata de procesos de búsqueda en los que cada nueva solución no es necesariamente mejor (en términos de la función objetivo que la inmediatamente anterior). Algunas veces aceptan, incluso, soluciones no factibles como paso intermedio para acceder a nuevas regiones no exploradas.

- Son relativamente sencillos; todo lo que se necesita es una representación adecuada del espacio de soluciones, una solución inicial (o un conjunto de ellas) y un mecanismo para explorar el campo de soluciones.
- Son generales. Prácticamente se pueden aplicar en la resolución de cualquier problema de optimización de carácter combinatorio. Sin embargo, la definición de la técnica será más o menos eficiente en la medida en que las operaciones tengan relación con el problema considerado.
- La regla de selección depende del instante del proceso y de la historia hasta ese momento. Si en dos iteraciones determinadas, la solución es la misma, la nueva solución de la siguiente iteración no tiene por qué ser necesariamente la misma

5.2.3. Aplicabilidad de los métodos heurísticos y meta heurísticos

Durante los recientes años, más y más algoritmos metaheurísticas han sido propuestos, extendiendo y diversificando la gama de métodos algorítmicos (Martí n.d.), lo que evidencia la amplia utilidad de estos métodos de optimización; de hecho, los algoritmos meta heurísticos son capaces de resolver problemas muy complejos de ingeniería cuyas soluciones no serían tan efectivas con métodos exactos, numerosas aplicaciones de estos algoritmos pueden ser encontradas en la literatura. (Merrikh-Bayat 2015), algunos ejemplos de ello son: solución de Sistemas de ecuaciones no lineales, agente viajero, planeación, despachos, diseño, planeación, automatización, etc. (Wang y Zhou 2014); inclusive, problemas de complejidad no Polinomial de alta dificultad computacional pueden ser resueltos por algoritmos meta heurísticos (Saji et al. 2014).

5.2.4. Taxonomía de los algoritmos meta heurísticos

En el presente punto analizaremos los diferentes sistemas de clasificación de los algoritmos meta heurísticos, haciendo énfasis en el sistema más utilizado y definiendo sus categorías.

5.2.4.1.Sistemas de clasificación de meta heurísticas:

Existen diversas maneras de clasificar a los algoritmos meta heurísticos, considerando entre otras cosas, su inspiración o la naturaleza de sus procedimientos (Martí n.d.). Además, (Blum y Roli 2003) señala formas alternativas de clasificación de los algoritmos, según realicen o no cambios en la función objetivo (funciones estáticas-funciones dinámicas), según usen o no memoria (corto plazo-largo plazo), según el número de estructuras vecinas (Una o varias) y según se basen en población o utilicen un único punto de partida (evolutivos o de trayectoria).

Según su inspiración, los algoritmos se dividen en: Algoritmos inspirados en la naturaleza y algoritmos no inspirados en la naturaleza (Lence 2007),de modo que los de abstracción natural se inspiran en leyes o principios naturales: inteligencia de enjambre, comportamiento de seres vivos (microorganismos, plantas y animales) (Melián et al. 2003) y los algoritmos de abstracción artificial (no inspirados en la naturaleza) se fundamentan en principios extraídos de las ciencias formales o estructuras creadas por los seres humanos. No obstante, (Lence 2007) afirma que dicha forma de clasificar los algoritmos no resulta conveniente, pues algunas veces dificultaría clarificar los atributos de los algoritmos en alguna de las dos clases, cuando estos usen principios de ambos tipos. De esta manera, optaremos por explicar más a fondo la clasificación de las metaheurísticas según la naturaleza de sus procedimientos, por ser este el sistema más ampliamente tratado

en la literatura, como se puede ver en (Lobo y Castro 2011), (Lence 2007), (Santana et al. 2004), (Pérez n.d.).

5.2.4.2.Clasificación de los algoritmos meta heurísticos según la naturaleza de sus procedimientos

Este sistema de clasificación se basa principalmente en el funcionamiento del algoritmo, y, básicamente existen 4 categorías: Meta heurísticas de relajación, constructivas, de búsqueda y evolutivas, aunque, cabe anotar que los algoritmos híbridos podrían formar una categoría aparte (Santana et al. 2004).

(García Sánchez n.d.) aporta la definición para cada clase de algoritmo:

- Meta heurísticas de relajación: Las metaheurísticas de relajación se refieren a procedimientos de resolución de problemas que utilizan relajaciones del modelo original (es decir, modificaciones del modelo que lo hacen más fácil de resolver), cuya solución facilita la solución del problema original.
- Las metaheurísticas constructivas: se orientan a los procedimientos que tratan de la obtención de una solución a partir del análisis y selección paulatina de las componentes que la forman.
- Las metaheurísticas de búsqueda: guían los procedimientos que usan transformaciones o movimientos para recorrer el espacio de soluciones alternativas y explotar las estructuras de entornos asociadas.
- Las metaheurísticas evolutivas: enfocadas a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones.

5.3. ANTECEDENTES

En el transcurso de los últimos cincuenta años, gran multitud de algoritmos meta heurísticos de optimización han sido desarrollados como respuesta a la necesidad de dar solución a una amplia gama de problemas complejos en diversas áreas del conocimiento, como lo señala (Merrikh-Bayat 2015): Durante las pasadas décadas, la naturaleza ha sido la fuente de inspiración para el diseño de nuevos algoritmos de optimización para problemas complejos de ingeniería.

De esta manera, abordaremos algunos de los principales trabajos realizados en este campo, siguiendo una secuencia cronológica.

Como lo indica (Merrikh-Bayat 2015), la primera aplicación fue en worksof Rechenberg 1965 (I. Rechenberg 1965). Trabajo que (Glover 1986) describe como la implementación de un experimento computacional, el cual se aplicó en el problema de traslación estable de la Royal Aircraft. Sin embargo, (Merrikh-Bayat 2015) aclara que el primer algoritmo general propuesto fue diseñado por Holland 1975 (Holland 1975b) y es conocido con el nombre de algoritmo genético (GA). GA fue pensado inicialmente como una Adaptación de sistemas naturales en sistemas artificiales, aplicando principios biológicos a través de la inteligencia artificial, simulando el proceso de evolución de las especies y aprovechándolo en optimización (Holland 1975a).

Después de una década (1983) (Kirkpatrick et al. 1983) Propone su "Simulated anneling" (SA) o "Recocido simulado" el cual es un algoritmo de búsqueda meta-heurística para problemas de optimización global; que realiza la búsqueda en un espacio bastante grande, este algoritmo se inspira en el proceso de recocido del

acero y de las cerámicas, una técnica que consiste en calentar y luego enfriar lentamente el material para variar sus propiedades físicas.

Según lo indica (Yazdani y Jolai 2015): En 1986 se propone el algoritmo Artificial Immune Systems, simulando el funcionamiento del Sistema inmunitario, empleándolo en la solución de problemas de optimización,.

En 1989 (Glover 1989) propone un algoritmo ampliamente difundido, conocido como "búsqueda tabú" (TS), el cual no se inspira en fenómenos naturales, sino que realiza la búsqueda marcando las soluciones ya revisadas, siendo este el primer algoritmo basado en memoria. Glover, publica nuevas mejoras a su algoritmo en 1990 (Glover 1990) y 1997 (Glover y Laguna 1997). Cabe anotar que con la primera versión de TS en 1989, Golver introduce el término Meta heurística (Santana et al. 2004).

El primer algoritmo en introducir inteligencia de enjambre fue el algoritmo "granja de hormigas" (ACO por sus siglas en inglés), propuesto por Dorigo M. en 1992 (Dorigo 1992), el cual simula el comportamiento de las hormigas en el proceso de acarrear su alimento. Luego de ACO han surgido gran cantidad de algoritmos que utilizan "evolución cultural" o inteligencia de enjambre, tal es el caso del conocido PSO (Enjambre de partículas) (Yazdani y Jolai 2015), propuesto por Eberhart en 1995(Eberhart 1995), basado en la estrategia de vuelo de las aves migratorias (Fukayama y Yoshida 2001), (Eberhart y Kennedy n.d.), PSO sería modificado posteriormente por (Eberhart y Xi 2001).

En 1997 (Storn y Price 1997) propone su algoritmo DE (evolución diferencial), como una meta heurística global para espacios continuos. (Abbass 2001) propone su MBO (marriage in honey bees optimizacion) inspirado en las abejas, empleando

inteligencia de enjambre para obtener sus soluciones. Tres años después, (2004), (Moscato y Cotta 2004) propone su "algoritmo mimético" (MA), que incluye auto adaptación, combinando búsqueda local y global en secuencias iterativas (Maier et al. 2014). Un año después, (Baudry et al. 2005) introduce su (BA) o "algoritmo bacteriológico".

Posteriormente, (2007), (F C Yang 2007) plantea "Flujo de agua", inspirado en el recorrido de las corrientes de agua en su descenso desde niveles más altos.

Como lo muestra (Bidar y Rashidy Kanan 2013), Yang introduce en 2008 su algoritmo "Luciérnaga", basado en inteligencia de enjambre e inspirado en el comportamiento de las luciérnagas, Un año más tarde (2009), Yang. (Arora y Singh 2013) expone "Búsqueda cucú" basado en la reproducción estratégica de los cucús. Algoritmo que sería mejorado por su diseñador en 2011 (Yang 2010).

En 2012 serían formulados "Water cycle", o "ciclo del agua", simulando dicho proceso natural (Eskandar et al. 2012) y Búsqueda de lobo "Wolf Search" (Tang et al. 2012), basado en la estrategia de caza de los lobos y empleando memoria a corto plazo; Un año después (2014), (Cheng y Prayogo 2014) formula su algoritmo bio-inspirado "búsqueda de organismos simbióticos", que simula las estrategias simbióticas de interacción adoptadas por los organismos para sobrevivir y propagarse en el ecosistema, también, (Kaveh y Mahdavi 2014) propone "Choque de cuerpos", algoritmo basado en colisiones unidimensionales entre los cuerpos, haciendo una analogía de leyes las leyes físicas. El mismo año, (Moosavian y Kasaee Roodsari 2014) postula su algoritmo "liga de fútbol", simulando la competición entre equipos en los torneos de futbol, dando cuenta de

la gran variedad de metáforas que se han empleado para el diseño de nuevos métodos de optimización.

En 2015, serían propuesta una amplia gama de algoritmos meta heurísticos, entre ellos: “Búsqueda estocástica de fractales” (SFS) (Salimi 2015), inspirado en las estructuras fractales, incluyendo implícitamente en sus parámetros las restricciones de los sistemas.

(LOA) “algoritmo león”, planteado por (Yazdani y Jolai 2015), inspirado en el comportamiento de los leones en la naturaleza; gradiente evolución (GE), diseñado por (Kuo y Zulvia 2015), haciendo una adaptación de los métodos exactos (LaGrange y PNL) para métodos aproximados; (BA) o “algoritmo murciélago”, propuesto por (Meng et al. 2015), inspirándose en el sistema de eco localización de los murciélagos; (Niu et al. 2015) plantea “mosca de la fruta” (FOA), inspirado en la búsqueda de alimento de las moscas.

Finalmente, en 2016 se propone el Algoritmo Eidético(Fong et al. 2016), basado en Wolf Search, introduciendo el concepto de memoria eidética para mejorar el algoritmo Búsqueda de lobo.

6. MATERIALES Y MÉTODOS

A continuación se expondrán los planteamientos hipótesis y el respectivo diseño metodológico de la presente investigación.

6.1. HIPÓTESIS

En este numeral se presentarán los tipos de hipótesis formuladas y las correspondientes variables involucradas en el planteamiento de las mismas.

6.1.1. Variables Involucradas

Previo al proceso de registro y medición de los datos, es necesario listar y definir las variables involucradas en el estudio:

- Algoritmo: El presente estudio tiene por objetivo determinar si el nuevo algoritmo propuesto representa mejoría con respecto a las lógicas más citadas en la literatura, de este modo, es imprescindible considerar como variable independiente la secuencia algorítmica, que, como entrada del sistema constará de varios niveles, uno por cada algoritmo de comparación además del nuevo método propuesto.
- Instancia de prueba: Los problemas que según la teoría son comúnmente empleados para probar métodos de optimización, representarán una variable independiente, que constará de tantos niveles como funciones objetivo se decida emplear para realizar la respectiva comparación entre algoritmos.

- Calidad de respuesta: Representa la proximidad de las soluciones con el valor óptimo, para cada función de prueba que arroje cada algoritmo de comparación, esta variable será una salida del sistema, pues dependerá de las variables "algoritmo" e "instancia de prueba"

Además, los parámetros de entrada inherentes a cada uno de los algoritmos, serán en cierto modo variables independientes para el estudio; entre ellos, serán comunes para todo algoritmo por lo menos: tamaño de la población inicial, dimensión de los vectores de variables de decisión y tiempo de cómputo.

6.1.2. Tipos de hipótesis a formular

En el presente trabajo se consideraron las siguientes hipótesis:

- Hipótesis de investigación: la hipótesis de investigación, busca dar respuesta a la pregunta de investigación, y aportar evidencia empíricamente que apoye o refute los supuestos que suscitaron inicialmente el estudio (Sampieri 2010), de este modo, nuestra hipótesis de investigación, en su definición conceptual quedaría de la siguiente manera:

  H_0: *"No existe diferencia significativa en el valor medio de la calidad de respuesta aportada por el nuevo algoritmo propuesto y el obtenido a través de las meta heurísticas más referenciadas".*

  H_1: Existe diferencia estadística significativa en el valor medio de calidad de respuesta aportada por el algoritmo propuesto, en comparación con las meta heurísticas más referenciadas.

  De rechazarse la hipótesis nula, se someterían a prueba las siguientes hipótesis:

H_0: *"La calidad de respuesta aportada por el nuevo método algorítmico propuesto NO representa una mejoría con respecto a la calidad alcanzada con los algoritmos meta heurísticos más referenciados en la literatura".*

H_1: *"La calidad de respuesta aportada por el nuevo método algorítmico propuesto representa una mejoría con respecto a la calidad alcanzada con los algoritmos meta heurísticos más referenciados en la literatura".*

- ANOVA unidireccional: aplicable para pruebas estadística que busquen analizar si más de dos grupos difieren significativamente entre sí en cuanto a sus medias y varianzas (Sampieri 2010). El presente trabajo usará este tipo de formulación en la determinación de la existencia de diferencias significativas en la media poblacionales de la variable calidad de respuesta de los algoritmos de comparación. El planteamiento de hipótesis será de la siguiente forma:

H_0: $\mu_i = \mu_j \; \forall i, j$

H_1: $\mu_i \neq \mu_j$ Para al menos un par i, j

A este análisis de varianza se le puede sumar una PRUEBA DE MÚLTIPLES RANGOS que realice la comparación pareada entre cada par i, j, especificando particularmente las comparaciones referentes al algoritmo propuesto, como se ilustra en (Kuo y Zulvia 2015), (Niu et al. 2015). De no cumplirse el supuesto de normalidad en los residuos, se empleará el equivalente no paramétrico de la prueba de múltiples rangos: la prueba de medianas de Mood.

Cabe señalar que los anteriores formatos de hipótesis, además de los métodos multivariados de regresión pueden ser empleados en la fase de ajuste de parámetros previa al experimento de comparación.

6.2. METODOLOGÍA

La presente investigación se rige por el siguiente diseño metodológico:

6.2.1. Tipo de investigación

En el presente estudio se busca establecer el nivel de eficiencia de un nuevo algoritmo meta heurístico de optimización global, probado inicialmente en problemas de variable real no restringidos. De esta manera, la investigación se enmarcará en un enfoque netamente CUANTITATIVO, por cuanto el nuevo método a evaluar supone la relación entre variables y amerita el uso de herramientas estadísticas que permitan establecer la existencia de diferencia significativa entre los valores de salida asociados a cada nivel de la variable independiente.

En cuanto al alcance de la investigación, y acorde con lo anteriormente expuesto, el presente trabajo puede ser considerada como un estudio CORRELACIONAL, pues busca medir el grado de relación entre dos o más variables, en concordancia con la definición que (Sampieri 2010) aporta sobre estudios correlaciónales. Sin embargo, el componente exploratorio está presente, pues se busca establecer una idea inicial sobre el desempeño y posibles potencialidades de un nuevo método de optimización.

La metodología a seguir en el transcurso de la investigación será EXPERIMENTAL, puesto que las condiciones del estudio permiten asegurar las especificaciones necesarias para considerarle un “Experimento puro” en el cual intervendrán múltiples variables explicativas y de respuesta en varios niveles, con un grupo de control y aplicando solo post prueba.

Consideraremos el presente trabajo como un estudio de FUENTE EMPIRICA, ya que los datos provienen de la experimentación en un ambiente controlado (Simulación por computadora).

6.2.2. Diseño experimental:

Según (Ferrer 2010), si en un estudio se manipulan variables independientes para verificar el efecto que esto tiene en variables dependientes bajo un ambiente controlado, la investigación se considera de tipo experimental; sin embargo, (CAMPBELL y STANLEY 1996) distingue entre pre experimentos, experimentos "puros" y cuasi experimentos. De manera que, para considerar un experimento como "puro" es necesario que este cumpla dos supuestos: aleatoriedad en la muestra y control sobre las variables independientes.

Para la presente investigación, las condiciones experimentales se configuran de modo que la secuencia de prueba de las instancias será aleatoria (generada a partir de una distribución uniforme) para cada algoritmo de comparación, por lo que se garantizará el principio de aleatoriedad, requisito indispensable para que el experimento sea considerado como "Experimento verdadero o puro", por otra parte, la manipulación de las variables independientes "Algoritmo" y "función de prueba", para el análisis de la variable dependiente "calidad de respuesta" nos permite afirmar que estamos ciertamente ante un experimento puro.

En cuanto al grado de manipulación de las variables, el presente trabajo considera varios grados para las dos variables independientes del sistema

(“Algoritmo” e “instancia de prueba”), pues habrá tantos niveles en “Algoritmo” como métodos de optimización se empleen en la comparación y tantos niveles en “instancia de prueba” como funciones objetivo en el experimento de comparación.

En el proceso de ajuste de parámetros, la realización de pre pruebas, anteriores a la manipulación de los valores de entrada de los algoritmos permitirán un marco referencial en la determinación de mejoría o retroceso en la búsqueda de la combinación optima de los parámetros para cada función objetivo. Sin embargo, el experimento de comparación solo requerirá de post prueba (al finalizar cada ejecución).

Para el tipo de experimento que atañe a la presente investigación, un grupo de control sería necesariamente un conjunto de soluciones generadas en forma aleatoria durante el tiempo promedio de ejecución de los algoritmos de comparación para cada instancia de prueba.

6.2.3. Muestra De Estudio

Para el caso específico de este estudio, la medición corresponde con la recolección de los datos arrojados por los algoritmos de optimización implementados en el lenguaje MATLAB, la variable “Calidad de respuesta” se registrará para cada corrida, tomando tantas mediciones como los parámetros estadísticos determinen como tamaño de muestra representativo de la población.

6.2.4. Recolección De Datos

- Indicador para la medición de la variable de respuesta: En la medición de la calidad de respuesta, se propone la utilización del indicador F.O. (Valor de la función objetivo) como lo sugieren (Eskandar et al. 2012), (Kuo y Zulvia 2015), (Yang 2010), (Niu et al. 2015), (Kaveh y Khayatazad 2012). Especificando el valor óptimo para cada instancia de prueba. Además, para el análisis inferencial se utilizará el indicador Z (puntaje típico estandarizado), que permite llevar a cabo una comparación global de la calidad de respuesta, sin recurrir a la verificación instancia por instancia.

  Cabe mencionar que el tiempo de cómputo se ve influido por las características de la computadora en la cual se realice la ejecución, por lo cual es necesario especificar dichas características y garantizar la homogeneidad de las mismas para la realización del experimento (Wang y Zhou 2014), con ello en mente, se empleará la función TIC-TOC del lenguaje MATLAB para establecer un tiempo constante como límite de computo, que servirá como estándar, evitando sesgos en la comparación, pues todos los algoritmos correrán durante la misma cantidad de tiempo.

- Parámetros de ejecución: según lo muestran (Kuo y Zulvia 2015) y (Hajipour et al. 2012), es necesario que todos los algoritmos corran en igualdad de condiciones con respecto a los parámetros que les son comunes (Tamaño de población inicial, numero de variables de decisión, tiempo de ejecución).

  Como se puede apreciar en la literatura, el número de algoritmos de comparación puede ser bastante reducido (2) (Yang 2010) o incluso superar

los veinte (Eskandar et al. 2012), sin embargo, es común que se empleen entre 4 (Kaveh y Khayatazad 2012), (Kuo y Zulvia 2015) y 6 algoritmos de comparación (Niu et al. 2015). Para nuestro estudio se decide realizar la comparación con los 4 algoritmos más referenciados en la literatura (algoritmo genético, enjambre de partículas, evolución diferencial, Recocido simulado) además de comparar con un algoritmo recientemente diseñado (eWSA) y con un algoritmo de generación aleatoria a modo de grupo de control.

En cuanto al número de funciones de prueba, estas oscilan entre las 10 (Yang 2010) y 40 funciones (Kuo y Zulvia 2015). Para el presente estudio se ha determinado emplear 20 instancias. Para determinar el número de iteraciones por ejecución, vemos que (Niu et al. 2015) emplea tan solo 500, mientras que (Kuo y Zulvia 2015) emplea hasta 3000 iteraciones, para nuestro caso se el número de iteraciones será determinado por el tiempo máximo de computo. La población inicial, se establecerá en 100 individuos (Kaveh y Khayatazad 2012) y el número de variables de decisión se dispondrá en varios niveles (Kuo y Zulvia 2015) (2, 5, 10, 20, 50,)

Se pretende calcular a la variable dependientes su valor de media y desviación estándar, aunque de ser necesario se obtendrá también el valor de la mediana.

6.2.5. Análisis De Datos

En el presente estudio se utilizarán técnicas de estadística inferencial, puesto que se realizarán pruebas de hipótesis pro pendientes por llegar a conclusiones aplicables a una población de datos a partir de una muestra representativa como se ha visto en la totalidad de las fuentes consultadas referentes al diseño

de meta heurísticas. Se emplearán análisis de varianza y pruebas de múltiples rangos-prueba de medianas de Mood para determinar entre qué pares de algoritmos de comparación existe o no diferencia significativa para la variable de salida en su valor medio, lo anterior en cada instancia de prueba; para ello, será necesario recurrir en primer lugar a la estadística descriptiva como paso previo a la inferencia estadística.

Las pruebas de hipótesis que tendrían lugar serían:

- Ajuste de distribución (Shapiro-Wilk)
- ANOVA unidireccional
- prueba de múltiples rangos o en su defecto Prueba de medianas de Mood

Trabajando con un 95% de confianza y un error del 5%.

Se considerará además el cálculo del número de funciones para las cuales los algoritmos alcanzaron el mejor resultado (Kuo y Zulvia 2015) y para cuales el óptimo (Yang 2010).

7. RESULTADOS Y DISCUSIONES

7.1. SELECCIÓN DE LAS INSTANCIAS DE PRUEBA:

Partiendo de la revisión bibliográfica, se hallaron en (Lobo y Castro 2011), (Fong et al. 2016), (Tang et al. 2012), (Kuo y Zulvia 2015), (Hajipour et al. 2012), (Arora y Singh 2013) las siguientes funciones de prueba empleadas en comparación de algoritmos de codificación real:

Tabla 1. Funciones de prueba empleadas en literatura revisada para experimentos de comparación.

Acley	Generalized Penalized2	Neumaier 3	Schwefel2.3
Aluffi	Goldstein y Price	Noncontinuous Rastrigin	Shaffer 1
Becker	Griewank	Paviani	Shaffer 2
Bohachevsky1	Levyymontalvo1	Periodic	Sphere
Bohachevsky2	Levyymontalvo2	Powell's Quadratic	Step Function
Camelback3	Michaelewicz	Rastrigin	Sum Of Different Power
Camelback6	Miele y Cantrell	Roenbrock	Weighted sphere
Dekkersyaarts	Modified Rosenbrock	Salomon	Whitley
Easom	Moved axis	Schwefel	Wood's
Exponential	Multi-gaussian	Schwefel1.2	Zhakarov
Generalized Penalized1	Neumaier 2	Schwefel2.2	

De la anterior lista, se seleccionaron en forma aleatoria 20 funciones, la cuales se enumeran a continuación:

Tabla 2. Funciones de prueba seleccionadas para el experimento de comparación

Funciones bidimensionales	Rótulo	Funciones n dimensionales	Rótulo
Aluffi	1	Rastrigin	10
Bohachevsky1	2	Acley	11
Bohachevsky2	3	Exponential	12
Camelback6	4	Rosenbrock	13
Camelback3	5	Salomon	14
Modified rosenbrock	6	Sphere	15
Periodic	7	Wighted sphere	16
Schaffer 1	8	Sum of different power	17
Schaffer 2	9	Step function	18
		Schwefel 2.2	19
		Griewank	20

En los anexos 1 y 2 se muestran las expresiones matemáticas y se ilustran las representaciones tridimensionales de las funciones de prueba seleccionadas para la comparación, construidas con la herramienta SURF del lenguaje MATLAB

7.2. SELECCIÓN DE LOS ALGORITMOS DE COMPARACIÓN:

En aras de seleccionar los algoritmos El proceso de escogencia de los algoritmos de comparación empleado en la validación del nuevo algoritmo propuesto siguió la secuencia descrita a continuación:

Muestra piloto: A partir de la revisión bibliográfica, se listaron los algoritmos usados en la comparación para el proceso de validación de nuevas meta heurísticas de codificación real. En total se revisaron 13 trabajos recientes para dicha finalidad, a saber: CROMÁTICO(Lobo y Castro 2011), DE(Storn 1997), eWSA(Fong et al. 2016), WSA(Tang et al. 2012), ELPSO(Jordehi 2015), SFS(Salimi 2015), GE(Kuo y Zulvia 2015), WCA(Eskandar et al. 2012), DFOA(Niu et al. 2015), BA(Yang 2010), LOA(Yazdani y Jolai 2015), ODMA(Hajipour et al. 2012), JFFA(Bidar y Rashidy Kanan 2013).

A continuación, se presentan los algoritmos empleados en los experimentos de comparación de las meta heurísticas anteriormente referenciadas:

Tabla 3. Algoritmos de comparación empleados en literatura revisada

Cromático	DE	eWSA	WSA	ELPSO	SFS	GE	WCA	DFOA	BA	LOA	ODMA	JFFA
GA	ANM	WSA	PSO	PSO	PSO	PSO	HM	PSO	PSO	IWO	GA	FFA
SA	aSA	ACO	ACO	GA	CS	DE	ASCHEA	GA	FPA	BBO	PSO	
TS				HS	MCS	ABC	IGA	DE	NBA	GSA		
				FSO	SA	GA	GA	FOA	DE	BA		
				SA	ABC		SAPF	IFFO		WWO		
				BSOA			SR	CLPSO		HuS		
				ABC			HS	GL				
							DE					
							CULDE					
							PSO					
							SMES					
							DEDS					
							HEAA					
							ISR					
							ABC					

La siguiente tabla muestra las respectivas frecuencias de cada uno de los algoritmos listados previamente, según el número de veces que fueron usados en experimentos de comparación en la literatura consultada:

Tabla 4. Frecuencias algoritmos de comparación

PSO	GA	DE	ABC	SA	WSA	ACO	HS	BA
8	6	5	4	3	2	2	2	2
cromático	TS	ANM	aSA	eWSA	ELPSO	FSO	BSOA	SFS
1	1	1	1	1	1	1	1	1
CS	MCS	GE	JFFA	FFA	ODMA	FPA	NBA	LOA
1	1	1	1	1	1	1	1	1
IWO	BBO	GSA	WWO	HuS	DFOA	FOA	IFFO	CLPSO
1	1	1	1	1	1	1	1	1
GL	WCA	HM	ASCHEA	IGA	SAPF	SR	CULDE	SMES
1	1	1	1	1	1	1	1	1
DEDS	HEAA	ISR						
1	1	1						

Posteriormente se consultaría el número de citas que alcanza cada algoritmo (haciendo uso de la herramienta Google académico “Ver anexo 5”), seleccionando de entre ellos, los

4 algoritmos más citados para emplearlos en la validación del nuevo algoritmo propuesto, a saber: GA, PSO, DE y SA.

Adicionalmente, se consideró incluir en la comparación al algoritmo eWSA por ser uno de los métodos más recientes, esencialmente para proporcionarle mayor vigencia al proceso de validación, puesto que los algoritmos más referenciados revisten ya cierta antigüedad, además de contar eWSA con una inspiración similar a la del nuevo algoritmo propuesto ESSA.

En aras de contar con un grupo de control para la realización del experimento, se sumó en la comparación un algoritmo de generación aleatoria, completando así 6 algoritmos de contrastación.

7.3. DESCRIPCIÓN ALGORITMOS DE COMPARACIÓN

A continuación, se explica la lógica del nuevo método meta heurístico propuesto (ESSA), así como un breve abordaje a los algoritmos seleccionados para el experimento de comparación, (La clasificación taxonómica, así como la estructura y secuencia lógica de los algoritmos de comparación se incluye en los anexos 3 y 4).

7.3.1. Nuevo algoritmo de Estrategias de Escape (Escape Strategies Algorithm [ESSA])

El nuevo algoritmo ESSA es una meta heurística de codificación real bío-inspirada, que simula la interacción entre un depredador y una manada de presas para realizar una búsqueda eficiente de soluciones en una región factible aprovechando la inteligencia de enjambre. (Algunas meta heurísticas como LOA, WSA y eWSA ya habían explorado el potencial de analogías similares). La analogía natural emula la forma como las presas tratan de huir de un depredador, y al mismo tiempo buscan mantener

la formación de la manada; de igual manera, el depredador intentará seguir los movimientos de la manada. Cada individuo durante la cacería empleará en distinta medida maniobras de evasión tales como el zigzagueo o los saltos (esto último en momentos avanzados de la cacería), fijando diferentes prioridades ya sea en la realización de dichas maniobras o en mantener la formación de la manada, en la pugna por hacerse con la posición más segura (óptimo).

Este algoritmo combina múltiples operadores que usan transformaciones vectoriales (Mostrando alguna similitud con métodos como PSO y DE) para realizar la búsqueda de vecinos y cubrir la región factible, valiéndose de la inteligencia de enjambre que permiten explorar de forma más eficiente, pues se emplea un cumulo de partículas realizando búsqueda simultánea en lugar de un solo individuo recorriendo una trayectoria, cubriendo así un espacio más amplio de soluciones.

7.3.1.1. Categorización taxonómica de ESSA:

El nuevo algoritmo de Estrategias de Escape, por su inspiración se clasifica como una meta heurística bio – inspirada, por cuanto emula una dinámica biológica como lo es la interacción entre seres vivos; según el número de estructuras vecinas, se clasifica como un algoritmo de múltiples estructuras, puesto que usa inteligencia de enjambre, es decir, una población de partículas que se cubren diferentes trayectorias dentro de una región factible en lugar de una partícula única. Según se realicen o no cambios en la función objetivo, ESSA se circunscribe en la categoría de las meta heurísticas de función estática, pues no modifica la función objetivo para realizar la búsqueda de soluciones. Según el uso de memoria, el algoritmo de Estrategias de Escape no hace un uso exhaustivo de memoria, por lo que se podría clasificar como una meta heurística

con memoria a corto plazo; finalmente, por la naturaleza de sus procedimientos, el algoritmo ESSA encaja en la categoría "algoritmo de búsqueda", puesto que emplea transformaciones (movimientos) para explorar la región factible en busca de soluciones y explotar las vecindades en busca de óptimos locales. Adicionalmente, ESSA se considera una meta heurística pura, por cuanto no se construye a partir de la combinación de los operadores de otras meta heurísticas, de manera que no puede ser clasificada como hibrida.

7.3.1.2. Notación:

Sea la matriz m, que denotará a la manada, de modo que m_{ij} representa la j-ésima variable de decisión correspondiente al i-ésimo individuo de la población, con $i \in \{1,2,3, \dots, n\}$, siendo n el tamaño de la manada y $j \in \{1,2,3, \dots, d\}$ donde d será la dimensión

Sean Li y Ls los límites inferior y superior de las variables de decisión.

$Posdep \in [0,1]$ Determinará la posición inicial del depredador.

Sea F.O. la función objetivo, entendida como: Seguridad de la posición de un individuo.

Sean vd e iml los vectores que contienen respectivamente las coordenadas de la posición del depredador y del Individuo mejor localizado en la manada. El mejor valor global ($mejv$) se entenderá como la Posición Más Segura.

7.3.1.3. Parámetros del algoritmo:

A continuación se enumeran las entradas que requiere el algoritmo ESSA:

Tabla 5. Entradas básicas ESSA

Entradas básicas
f: función objetivo
d: dimensión (Número de variables de decisión)
n: Tamaño de la población (manada)
$Stoping\ criteria$: Criterio de parada

Tabla 6. Parámetros de funcionamiento ESSA

Parámetros de funcionamiento	
h_1: proporción en que se dará preponderancia a correr en la dirección perpendicular a la dirección del depredador (Zigzagueo)	k_1: proporción en que el depredador dará preponderancia a seguir al iml
h_2: proporción en que se dará preponderancia a correr en la dirección promedio del grupo (Mantener la formación)	k_2: proporción en que el depredador dará preponderancia a seguir la dirección promedio de la manada (Anticipar movimientos)
h_3: proporcion en que se dará preponderancia a correr en la dirección del individuo mejor localizado (Buscar posición más segura)	k_3: Proporción en que el depredador dará preponderancia a seguir la posición promedio de la manada (persecución)
Posdep: Número real entre 0 y 1 que determinará la posición inicial del depredador en la región factible	Jumper criteria: Criterio de cumplimiento del tiempo o número de iteraciones mínimo para permitir empleo de operador de saltos
$\sum_{i=1}^{3} h_i \approx \sum_{i=1}^{3} k_i \approx 1$	

7.3.1.4. Operadores:

A continuación se listan los operadores empleados en Escape Strategies Algorithm:

Tabla 7. Operadores ESSA

Entenderemos V(d+1) como el valor en F.O. de un vector V cualquiera		
OPERADOR	**DEFINICIÓN**	**ESTRUCTURA**
(Aclaración: se asigna el valor de la F.O. en la columna d+1 de cada individuo)		
Calcular holguras	Calcular la longitud máxima del movimiento de cada individuo para mantener las variables de decisión dentro de las fronteras de la región factible, hq: es la matriz de holguras por exceso y hl la matriz de holguras por defecto	hq=ls*ones(n,d+1)-m; hl=li*ones(n,d+1)-m;

Comenzar huida de la manada	La manada detecta la presencia del depredador e inicia la huida.
	Cada individuo de la manada toma una posición vecina aleatoria (manteniendo los límites demarcados por las holguras).
	Sea $mmov$ la matriz que contiene las nuevas posiciones luego del movimiento y
	$mdir$ la matriz que contiene la dirección del movimiento
		`mdir=inf(n,d+1) Inicializar mdir`
		
		`for x=1:n`
		`  for y=1:d+1`
		`mdir(x,y)=hl(x,y)+(hq(x,y)-hl(x,y))*rand`
		`  end`
		`end`
		`mmov=m+mdir`

Reorientar depredador	Calcular los puntos de referencia que orientan los movimientos del depredador: vectores de dirección y posición promedio ($vecdir, vecpos$)	`Vecdir,vecpos=null`
		`for y=1:n`
		`vecdir(y)=sum(mdir(1:n,y))/n;`
		`vecpos(y)=sum(mmov(1:n,y))/n;`
		`end`

Calcular individuo mejor localizado	El individuo mejor localizado, lo seleccionamos, comparando al mejor individuo de la población inicial (m), con el mejor individuo de la población luego de mover la manada ($mmov$), el que posea una mejor evaluación en la función objetivo será el iml	`[~,posiml1]=min(mmov(:,d+1));`
		`iml1=mdir(posiml1,:);`
		`[~,posiml2]=min(m(:,d+1));`
		`iml2=m(posiml2,:);`
		`if iml1(d+1)<iml2(d+1)`
		`  iml=iml1;`
		`else`
		`  iml=iml2;`
		`end`

Mover depredador	El depredador se mueve a través de la operación vectorial vd=vd+k1(iml-vd)+k2(vecdir-vd)+k3(vecpos-vd);	`vd=vd+k1*(iml-vd)+k2*(vecdir-vd)+k3*(vecpos-vd)`

Perp (Crear vector de Zigzagueo)	Crear vector perpendicular a la resta vectorial entre cada individuo y el depredador, aquí se propone una alternativa de zigzagueo en ángulo de 90° que crea vectores aleatorios $perp$ que cumplen que la condición matemática de ortogonalidad, es decir, que el producto vectorial entre un X dado y $perp$ sea igual a cero.
	
	Esta función de zigzagueo tiene como parámetros un vector v y vd y solo se usa dentro del operador "Crear parámetros de movimiento para la manada"
	
	Se proporciona el código en lenguaje MATLAB.
	Nota:
		`function[vec]=perp(d,vd)`
		`cg=randi(d);`
		`vgrup1=randperm(d,cg);`
		`vgrup2=inf(1,d-cg);`
		`fg=randperm(d);`
		`a=vd;`
		`v1=inf(1,cg);`
		`for x=1:cg`
		`v1(x)=a(vgrup1(x));`
		`end`
		`gg=0;`
		`for x=1:d`
		`band=0;`
		`for y=1:cg`
		`  if fg(x)==vgrup1(y);`
		`  band=1;`
		`  end`
		`end`
		`if band==0`
		`gg=gg+1;`
		`vgrup2(gg)=fg(x);`
		`end`

	"Randperm(b)" genera permutaciones del vector [1,2,…,b]
	
	Randi(b,[dimx,dimy]): Genera vectores de tamaño dimxXdimy, que contienen números aleatorios enteros entre 1 y b.
	
	Sum(X) suma las posiciones de un vector X dado.
	
	Inf(dimx,dimy), Ones(dimx,dimy), Zeros(dimx,dimy) generan vectores de tamaño dimxXdimy cuyas posiciones son respectivamente Infinitos, unos o ceros
		end
		v2=inf(1,gg);
		for x=1:gg
		v2(x)=a(vgrup2(x));
		end
		sum1=0;
		sum2=0;
		for x=1:cg
		sum1=sum1+v1(x);
		end
		for x=1:gg
		sum2=sum2+v2(x);
		end
		k=rand;
		if not(sum1==0)
		fact=-k*(sum2/sum1);
		else
		fact=0;
		end
		vpm=inf(1,cg);
		vsm=vpm;
		for j=1:cg
		vpm(vgrup1(j))=1;
		vsm(vgrup1(j))=0;
		end
		for j=1:gg
		vpm(vgrup2(j))=0;
		vsm(vgrup2(j))=1;
		end
		vec=fact*vpm+k*vsm;
		vec(d+1)=inf;

Reorientar manada	Calcular los puntos de referencia que orientan el siguiente movimiento de la manada, asignando a una matriz el respectivo vector de zigzagueo de cada individuo (mdp), crear matrices $mvecdir$ y $mdirml$ que contienen en cada fila réplicas de los vectores de posición promedio de la manada y del iml respectivamente, la función de movimiento depende de estos valores.	Inicializar:
		mdp=inf(n,d+1);
		for i=1:n
		mdp(i,1:d+1)=perp(v,vd);
		end
		Inicializar:
		mvecdir,mdiriml=inf(n,d+1)
		for x=1:n
		mvecdir(x,:)=vecdir
		mdiriml(x,:)=iml
		end

Mover manada	La manada se mueve de acuerdo a la operación vectorial m=mmov+h1*(mdp-mmov)+h2*(mvecdir)+h3*(mdiriml-mmov)	mmov=mmov+h1*(mdp-mmov)+h2*(mvecdir)+h3*(mdiriml-mmov) m=mmov
Saltos	En fases avanzadas de la cacería, los individuos pueden dar saltos para alejarse bruscamente del depredador o para acercarse a una posición más segura de forma más rápida, el operador de	Minimo= F.O. para mejorglobal minpos=posicion i del mejor individuo if iteraciones>1

	saltos aquí propuesto elige al azar un individuo de la población y reemplaza uno o más de sus genes (coordenadas de su posición), por el valor en la función objetivo del mejor individuo, siempre y cuando dicho valor se encuentre dentro de los límites de la región factible y solo sí dicha operación mejora la solución actual. Para nuestro caso se escogió un reemplazo de d posiciones	`if and(minimo>=li,minimo<=ls) if minpos<n-1 m(minpos+1,:)=ones(1,d+1)* minimo else m(minpos-1,:)=ones(1,d+1)*minimo end end end`
Actualizar iml y mejv	Se calcula el mejor valor de la población actual y se compara con el valor en la función objetivo del mejor global, y si este mejora la solución actual, el nuevo valor será asignado a iml y al mejor global, luego el mejor global se contrasta con el valor en F.O del depredador (vd) y si este mejora la solución esta se reemplaza por el valor de vd.	`[minimo,minpos]=min(m(:,d+1));if m(minpos,d+1)<mejv(d+1) iml=m(minpos,:) mejv=iml; end if vd(d+1)<mejv(d+1) mejv=vd iml=vd end`
Actualizar parámetros	En el presente algoritmo, los parámetros son dinámicos, es decir, h y k pueden cambiar sus valores para favorecer la exploración en las fases iniciales de la cacería o la explotación en las iteraciones finales, aquí se muestra un método que convierte los parámetros en función del tiempo o del número de iteraciones. t: tiempo o iteración actual, $tmax$: tiempo límite o número máximo de iteraciones	`h1=(t/tmax) h2=(1-(t/tmax))/2 h3=h2`

7.3.1.5. Secuencia lógica de ESSA:

7.3.2. Optimización Por Enjambre De Partículas (PSO)

Tabla 8. Estructura de código ESSA

Entenderemos V(d+1) como la evaluación en F.O. de un vector v cualquiera, la notación V(i,:) se entenderá como [v1,v2,…,vd]

INICIO

- Inicializar parámetros: f, d, n, li, ls, h1, h2, h3, k1, k2, k3, stoping criteria, jumper time;

  Crear población inicial:
- m=li*ones(n,d+1)+(ls-li)*rand(n,d+1);
- Evaluar(m)
- mejv=Individuo_con_mejor_evaluacion_en_m
- Calcular_holguras
- Comenzar_huida_de_la_manada
- Evaluar(mmov)

  Inicializar depredador:
- Vd= vd=li*ones(1,d+1)+posdep*(ls-li); Posdep $\in \mathbb{R}$ [0,1]
- Evaluar(vd)
- Calcular_individuo_mejor_localizado

  Mientras no se cumpla Stoping criteria
  - Reorientar depredador
  - Mover_depredador
  - Reorientar manada
  - Mover_manada
  - Evaluar(vd)
  - Evaluar(m)

    Si se cumple jumper criteria
    - Saltos

    Finsi
  - Actualizar_iml_y_mejv
  - Actualizar_parametros

  Finmientras

- Mostrar Mejv
- **FIN PROCESO**

Planteado por (Eberhart 1995) y modificado por (Eberhart y Xi 2001), es una poderosa meta heurística de codificación real que emplea inteligencia de enjambre para realizar la búsqueda en la región factible. PSO simula la estrategia de vuelo de las aves migratorias, empleando operaciones vectoriales para emular el movimiento de las partículas (Individuos de la parvada). La inspiración y características de PSO se exponen más ampliamente en (Eberhart 1995), (Eberhart y Kennedy n.d.), (Eberhart y Xi 2001),

(Lessmann et al. 2011). En el presente trabajo se implementó la versión de PSO propuesta por (Eberhart y Xi 2001)

7.3.3. Algoritmo genético (GA)

El algoritmo genético, propuesto por (Holland 1975b), se constituye uno de los algoritmos más conocidos y más usados, considerándose además como la primera meta heurística propiamente dicha (Zanakins, S. H. y Evans 1981); Genetic algorithm (GA) simula el proceso de selección natural para optimizar un conjunto de soluciones iniciales (individuos) a través de una secuencia lógica que incluye operadores de selección, cruce, mutación y relevo generacional. GA se expone en forma más amplia en (Holland 1975b), además, (Chelouah y Siarry 2000) plantea una versión para espacios continuos y (Houck y Kay 2008) una guía para su implementación, en este estudio se empleó la versión de GA expuesta en (Figielska 2009), por ser una versión simple y actualizada.

7.3.4. Evolución diferencial (DE)

El algoritmo de evolución diferencial (DE por sus siglas en inglés <<Differential evolution>>) Fue por (Storn y Price 1997), es un algoritmo evolutivo que realiza búsquedas de vecindades a través de operaciones vectoriales (Calculando Deltas o diferenciales) entre valores escogidos aleatoriamente de entre una población inicial, y sumando dicho delta a un vector transitorio luego de re escalar el valor diferencial; el vector transitorio, de mejorar el valor inspeccionado en la matriz procederá a reemplazarlo; dicho proceso se repite para cada individuo en la matriz y para un determinado número de iteraciones. DE se expone con mayor detalle en (Storn y Price 1997) y en DFOA (Niu et al. 2015), el cual se basa en evolución diferencial. En este trabajó se implementó la versión de DE propuesta por (Storn 1997)

7.3.5. Recocido simulado (SA)

El algoritmo de recocido simulado es una meta heurística de inspiración natural que simula el proceso físico de la cristalización, producto de calentar solidos a altas temperaturas y el posterior enfriamiento hasta lograr estados de baja energía; este algoritmo fue propuesto por (Kirkpatrick et al. 1983). SA se caracteriza por aceptar soluciones que empeoran la mejor actual con tal de escapar de óptimos locales, Simulated Annealing es expuesto en mayor detalle en (Kirkpatrick et al. 1983) y (Mousavi et al. 2013).

7.3.6. Eidético (eWSA)

La meta heurística EIDETIC WOLF SEARCH ALGORITHM (eWSA), propuesta por (Fong et al. 2016), es uno de los algoritmos más recientemente diseñados, con una analogía natural similar a la propuesta en ESSA, eWSA fue planteado como una modificación de WOLF SEARCH ALGORITHM (WSA), incluyendo la innovación de memoria eidética a la lógica del algoritmo "Búsqueda de lobo" original, simulando la estrategia de caza de los lobos. WSA y eWSA se explican más ampliamente en (Tang et al. 2012) y (Fong et al. 2016). En el presente trabajo se implementa la versión de eWSA propuesta en (Fong et al. 2016).

7.3.7. Algoritmo de generación aleatoria (RANDOM)

Para la presente investigación se consideró incluir un algoritmo de generación aleatoria de soluciones, en aras de que este actúe como grupo de control, mostrando la diferencia en la calidad de respuesta de las meta heurísticas de comparación y la generación arbitraria de soluciones. El algoritmo de generación aleatoria no puede considerarse como una meta heurística, pues no sigue ninguna lógica que implique el uso de inteligencia artificial, sino que solamente fabrica soluciones al azar (generalmente sigue una distribución uniforme);

RANDOM (Término en inglés) es el método aproximado más simple de exploración de regiones factibles.

7.4. PARAMETRIZACIÓN DE ALGORITMOS DE COMPARACIÓN

Para llevar a cabo el proceso de parametrización, los métodos usados incluyen técnicas de diseño experimental , y métodos algorítmicos de configuración de parámetros (López-ibáñez et al. 2016).

Según lo afirma (López-ibáñez et al. 2016) Los algoritmos de optimización modernos típicamente requieren de un gran número de parámetros a optimizar; la meta inmediata es buscar automáticamente la mejor configuración de parámetros de un algoritmo de optimización. además, como lo enuncia (Krus y Andersson 2016): los trabajos en esta área han demostrado que la optimización puede ser usada en el ajuste de parámetros.

(López-ibáñez et al. 2016) también afirma que, últimamente la configuración automática de algoritmos ha mostrado el potencial para imponer un nuevo paradigma de optimización. Es por ello que en la presente investigación se optó por emplear un método de parametrización no estadística, que aprovecha el potencial de los algoritmos meta heurísticos en la optimización de otros algoritmos, es decir, convirtiendo la calidad media de respuesta de un algoritmo a optimizar en la función objetivo de otro algoritmo, tomando la configuración de parámetros como variables decisión que deben ser optimizadas.

En el proceso de parametrización, se empleó Particle Swarm optimization (PSO) para optimizar los parámetros de los demás algoritmos y a sí mismo, pues su estructura simple, además de permitir una fácil y rápida adaptación para los fines requeridos, alcanza un mayor número de iteraciones por unidad de tiempo que las lógicas de estructura más compleja. (Como se evidencia en el ANEXO 9 “tabla resumen iteraciones/tiempo vs

algoritmo") Cabe anotar, que en la calibración, PSO partió de la base de soluciones previamente generadas por una heurística de generación aleatoria y no de soluciones arbitrarias para mejorar su rendimiento.

7.4.1. Adaptación de PSO para la optimización de algoritmos:

Para la tarea de ajuste de parámetros fue necesario realizar las siguientes adaptaciones al algoritmo Enjambre de partículas:

- Reemplazar la función de evaluación por la función "llamar algoritmo", que invocaba a cada uno de los algoritmos a optimizar dependiendo de un índice: (1: ESSA, 2: PSO, 3: GA, 4: DE, 5: SA, 6: eWSA, 7: RANDOM), tomando como la correspondiente F.O. la calidad de respuesta media del algoritmo a optimizar para una determinada configuración.
- Reemplazar los valores de $m_{ij} \in [li, ls]$ que contenía valores de variables de decisión para una determinada F.O. por C_{ij}, que contiene el valor del j-ésimo parámetro de un algoritmo a optimizar de la i-ésima configuración listada, donde $i \in [1, n]$ con n el numero de configuraciones probadas por iteración y $j \in [1, npar]$ donde $npar$ corresponde al número de parámetros requeridos por el algoritmo a optimizar.
- La salida del PSO modificado sería la mejor configuración de parámetros hallada para el algoritmo a optimizar en un cierto número de iteraciones.

7.4.2. Configuraciones de parámetros seleccionadas para cada algoritmo:

A continuación se listan los valores arrojados por el proceso de ajuste de parámetros para cada algoritmo:

Tabla 9. Parámetros seleccionados para el experimento de comparación

ALGORITMO	NÚMERO DE PARÁMETROS	CONFIGURACIÓN
ESSA	[posdep,n,h1,h2,h3,k1,k2,k3,Jumper criteria] (7)	[1,100,↑,↓,↓,0 .3305, 0.3305, 0.3305,1]
PSO*	[n,w,c1,c2]	[100, ↓, ↑, ↓]
GA	[n,pm,Tor] (2)	[100,0.7177,8]
DE	[n,CR,F] (2)	[100,0.1,0.9]
SA	[n,t0,a] (2)	[100,401.5235,0.1268]
eWSA	[n,p,a,s,v,u,fdeholgura] (7)	[24,0.183875397,0.222477929, 0.784819997,0.254304697,75,0 .011444005]
Para el caso de ESSA, se consideró la posibilidad de usar parámetros estáticos o dinámicos, el algoritmo de parametrizacion arrojó una configuración mixta (estática-dinámica de los parámetros) (↑: $creciente\ 0 \rightarrow 1$, ↓ $decreciente\ 1 \rightarrow 0$), para lograr ese crecimiento y decrecimiento se expresaron los parámetros como función del tiempo, para que estos aumentaran o disminuyeran en la misma proporción en que avanzaba el tiempo de ejecución. Este tipo de configuración fue la utilizada en PSO		

7.5. ANÁLISIS DE RESULTADOS

A continuación, se explica el diseño experimental empleado para el experimento de comparación, así como los resultados arrojados por la comparación entre algoritmos, presentando tanto un análisis descriptivo como un análisis estadístico inferencial:

7.5.1. Diseño experimental:

El presente trabajo siguió un diseño experimental simple bajo la siguiente configuración:

Tabla 10. Diseño experimental

VAR 1	Var 2 (Calidad de respuesta)				
Algoritmo	Instancia	F.O. replica 1	F.O. replica 2	F.O. replica 3	F.O. replica 4
7 niveles	53 niveles	1 nivel	1 nivel	1 nivel	1 nivel

Donde las instancias de prueba se configuran de la siguiente forma:

Tabla 11. Instancias de prueba

Instancia	Función	Dimensión	Rotulo	Instancia	Función	Dimensión	Rotulo
1	1	2	D2F1	28	14	20	D20F14
2	2	2	D2F2	29	14	50	D50F14
3	3	2	D2F3	30	15	5	D5F15
4	4	2	D2F4	31	15	10	D10F15
5	5	2	D2F5	32	15	20	D20F15
6	6	2	D2F6	33	15	50	D50F15
7	7	2	D2F7	34	16	5	D5F16
8	8	2	D2F8	35	16	10	D10F16
9	9	2	D2F9	36	16	20	D20F16
10	10	5	D5F10	37	16	50	D50F16
11	10	10	D10F10	38	17	5	D5F17
12	10	20	D20F10	39	17	10	D10F17
13	10	50	D50F10	40	17	20	D20F17
14	11	5	D5F11	41	17	50	D50F17
15	11	10	D10F11	42	18	5	D5F18
16	11	20	D20F11	43	18	10	D10F18
17	11	50	D50F11	44	18	20	D20F18
18	12	5	D5F12	45	18	50	D50F18
19	12	10	D10F12	46	19	5	D5F19
20	12	20	D20F12	47	19	10	D10F19
21	12	50	D50F12	48	19	20	D20F19
22	13	5	D5F13	49	19	50	D50F19
23	13	10	D10F13	50	20	5	D5F20
24	13	20	D20F13	51	20	10	D10F20
25	13	50	D50F13	52	20	20	D20F20
26	14	5	D5F14	53	20	50	D50F20
27	14	10	D10F14				

El anterior diseño fue aleatorizado empleando una columna de valores provenientes de una distribución uniforme, y se replicó 4 veces según el cálculo del tamaño de la muestra (Anexo 11).

Luego de llevar a cabo la parametrización de los algoritmos de comparación, se procedió a llevar a cabo el experimento de comparación, en aras de contrastar la calidad de respuesta de los algoritmos

Los algoritmos fueron evaluados en 20 funciones restringidas, 9 de ellas bidimensionales y 11 de n variables, probadas en dimensiones de 5, 10, 20 y 50 variables (completando así 53 instancias de prueba) , Los 6 algoritmos meta heurísticos y el algoritmo de generación aleatoria fueron programados en el lenguaje MATLAB y ejecutados en un equipo de cómputo cuyas especificaciones se detallan en el ANEXO 12, con un tiempo estándar por corrida de 10 segundos, con el fin de minimizar las funciones de prueba, usando población estándar en los algoritmos de múltiples estructuras vecinas como lo sugieren (Kuo y Zulvia 2015) y (Hajipour et al. 2012) (dicho tamaño de población estándar se estipuló en 100 individuos, a excepción de eWSA, para el que la literatura sugiere poblaciones pequeñas, tal y como lo confirmó su parametrización). A continuación se presenta el resumen de los resultados obtenidos y su respectivo análisis descriptivo e inferencial.

7.5.2. Análisis descriptivo:

En el anexo 6 se presenta la tabla de resultados del experimento, (calidad de respuesta promedio, iteración vs algoritmo) la cual se resume a continuación:

Tabla 12. Resumen de resultados experimentales, número de instancias para las cuales cada algoritmo alcanza el mejor promedio muestral, discriminado por dimensiones

ALGORITMO	DIMENSIÓN					TOTAL
	2D	**5D**	**10D**	**20D**	**50D**	
ESSA	3	5	5	8	11	32
PSO	8	3	1	0	0	12
GA	0	1	1	0	0	2
DE	3	5	6	4	0	18
SA	0	1	1	0	0	2
eWSA	0	1	1	0	0	2
RANDOM	0	0	0	0	0	0

Como se puede apreciar en la anterior tabla, el algoritmo de Estrategias de Escape obtuvo el mejor promedio muestral para 32 de las 53 instancias de prueba, seguido de los algoritmos DE, PSO, eWSA, SA y GA con 18, 12, 2, 2 y 2 instancias respectivamente. De esta manera, se evidencia la gran consistencia en las soluciones aportadas por el algoritmo de Estrategias de Escape (ESSA). Nótese además que a medida que aumenta el número de variables de decisión (Dimensión) aumenta también el número de instancias para las cuales ESSA logra la mayor calidad de respuesta en comparación con los demás algoritmos; por lo que reviste un gran potencial para problemas de grandes dimensiones.

En el anexo 7 se presenta la tabla de resultados del experimento (Desviación estándar, instancia vs algoritmo), la cual se resume a continuación:

Tabla 13. Resumen de resultados experimentales, número de instancias para las cuales cada algoritmo alcanza la menor variación (desviación estándar muestral), discriminado por dimensiones

ALGORITMO	DIMENSIÓN					TOTAL
	2D	5D	10D	20D	50D	
ESSA	3	6	5	8	10	32
PSO	8	1	1	0	0	10
GA	0	1	1	0	0	2
DE	4	3	6	4	1	18
SA	0	1	2	0	0	3
eWSA	0	2	1	0	0	3
RANDOM	0	0	0	0	0	0

Como se puede apreciar en la anterior tabla, el algoritmo de Estrategias de Escape obtuvo el mejor promedio muestral para 32 de las 53 instancias de prueba, seguido de los algoritmos DE, PSO, eWSA, SA y GA con 18, 10, 3, 3 y 2 instancias respectivamente. De esta manera, se evidencia la gran consistencia en las soluciones aportadas por el algoritmo de Estrategias de Escape (ESSA). Nótese además que a medida que aumenta el número de variables de decisión (Dimensión) aumenta también el número de instancias para las cuales ESSA logra la menor variación en la calidad de respuesta en comparación con los demás algoritmos; por lo que muestra gran potencial para problemas de grandes dimensiones.

7.6. ANÁLISIS INFERENCIAL:

El indicador empleado para la comparación a través de técnicas inferenciales es el PUNTAJE TÍPICO ESTANDARIZADO (Z), $Z \sim N(0,1)$ entendido como $z = \frac{x_i - \bar{x}}{\sigma}$,

El cual permite realizar comparación entre instancias diferentes, dándonos la oportunidad de comparar resultados globales en lugar de verificar instancia por instancia. A

continuación se ilustra los resultados del análisis de varianza (ANOVA simple) para determinar si la variable "Algoritmo" tiene incidencia relevante sobre la variable "Calidad de respuesta" medida en términos del puntaje típico estandarizado con un 95% de confianza. La tabla ANOVA fue obtenida a través del software StatGraphics:

Tabla 14. ANOVA simple

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
Entre grupos	1128,58	6	188,096	639,25	0,0000
Intra grupos	434,599	1477	0,294244		
Total (Corr.)	1563,18	1483			

Como se puede observar, Valor P <0.05 por lo tanto se rechaza la hipótesis nula de igualdad de medias para Z entre un nivel de algoritmo y otro, por lo que se concluye, con un 95% de confianza que existe diferencia estadística significativa en la calidad de respuesta de los algoritmos de comparación.

Ahora, la verificación de la normalidad de los residuos, que determina el tipo de inferencia a realizar (paramétrica-no paramétrica), se llevó a cabo a través de la prueba de Shapiro-Wilk, (ver anexo 8). Rechazándose la hipótesis nula de normalidad de los residuos, por lo cual se omitió la verificación de los supuestos de independencia y homocedasticidad. De esta manera, se determinó el uso de estadística no paramétrica (más específicamente "Prueba de medianas de Mood") para el análisis inferencial de los resultados del experimento.

A continuación se presentan los resultados de la prueba de comparación de medianas de Mood, llevada a cabo con un 95% de confianza: (los índices de los algoritmos {1, 2, 3, 4, 5, 6, 7} corresponden respectivamente a ESSA, PSO, GA, DE, SA, eWSA, RANDOM)

Tabla 15. Prueba de medianas de Mood

algoritmo	*Tamaño de Muestra*	*n<=*	*n>*	*Mediana*	*LC inferior 95,0%*	*LC superior 95,0%*
1	212	157	55	-0,374643	-0,380816	-0,373424
2	212	84	128	-0,349951	-0,356279	-0,33897
3	212	137	75	-0,371743	-0,373799	-0,368772
4	212	147	65	-0,373695	-0,378867	-0,371581
5	212	137	75	-0,371575	-0,373367	-0,368888
6	212	78	134	-0,32547	-0,350086	-0,315146
7	212	5	207	2,5023	2,42962	2,55871

Estadístico = 333,434 Valor-P = 0,0
Total n = 1484
Gran mediana = -0,357495

Como se aprecia, Valor P <0.05, para la prueba de Mood, de esta manera, podemos rechazar la hipótesis nula de igualdad, y afirmar con un 95% de confianza que existe diferencia estadística significativa entre las medianas de “Calidad de respuesta (Z)” entre los distintos niveles de la variable “Algoritmo”. A continuación se presenta el grafico de cajas y bigotes de la comparación:

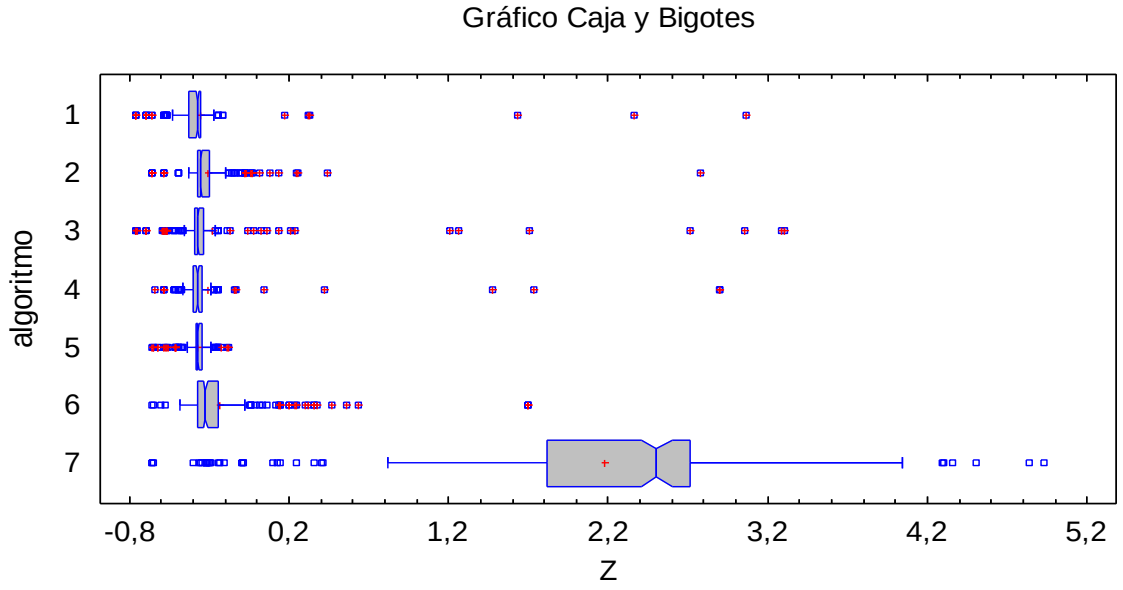

Grafico 1. Cajas y bigotes Z para algoritmo

Para mayor claridad en los resultados, se amplía la información de la Prueba de Mood para identificar las diferencias en la calidad de respuesta entre los diferentes niveles de la variable algoritmo, de modo que dos algoritmos tendrán una diferencia significativa si no existe intersección entre los intervalos de confianza para la mediana:

Tabla 16. Prueba de medianas de Mood con grupos homogéneos y frecuencias relativas

algoritmo		*Tamaño de Muestra*	*n<=*		*n>*		*Mediana*	*LC inferior 95,0%*	*LC superior 95,0%*	*Grupos homogéneos*
			Absoluto	*Relativo*	*Absoluto*	*Relativo*				
1	ESSA	212	157	74%	55	26%	-0,37464	-0,380816	-0,373424	X
2	PSO	212	84	40%	128	60%	-0,34995	-0,356279	-0,33897	XX
3	GA	212	137	65%	75	35%	-0,37174	-0,373799	-0,368772	XX
4	DE	212	147	69%	65	31%	-0,3737	-0,378867	-0,371581	X
5	SA	212	137	65%	75	35%	-0,37158	-0,373367	-0,368888	X
6	eWSA	212	78	37%	134	63%	-0,32547	-0,350086	-0,315146	X
7	RANDOM	212	5	2%	207	98%	2,5023	2,42962	2,55871	X

Como se puede observar, existen 4 grupos homogéneos (No existe diferencia estadística significativa entre los niveles de "Algoritmo" que tengan en común al menos una columna de X), de esta manera, los grupos homogéneos quedan conformados de la siguiente forma:

Tabla 17. Grupos homogéneos

Grupo Homogéneo	Algoritmos
1	(1,4,3) (ESSA, DE,GA)
2	(4,3,5) (DE,GA,SA)
3	(2,6) (PSO, eWSA)
4	7 (RANDOM)

La calidad de respuesta varía significativamente de un grupo homogéneo a otro, por lo que podemos afirmar que los algoritmos que alcanzaron una mejor calidad de respuesta en el presente experimento son, en orden: ESSA (Escape Strategies Algorithm), DE (Differential evolution) y GA (Genetic Algorithm), alcanzando el algoritmo ESSA los mejores valores tanto en mediana, como en límite inferior y límite superior del intervalo de confianza para la mediana, demostrando ser tan bueno en calidad de repuesta como los métodos más referenciados en la literatura científica e incluso superando a varios de ellos.

8. CONCLUSIONES

Como se puede ver en los resultados del experimento numérico, el algoritmo de Estrategias de Escape ha demostrado tener un gran potencial de optimización, constituyéndose así en una nueva y muy eficiente alternativa para la solución de problemas de optimización de variable real.

Este experimento adicionalmente, ejemplificó la aplicabilidad de los métodos meta heurísticos, logrando buenas soluciones en una amplia gama de instancias de prueba de configuraciones muy variadas y que presentaban obstáculos diferentes para su optimización.

Como podemos apreciar en los resultados del experimento, el grupo de control (RANDOM) fue superado por la totalidad de algoritmos meta heurísticos, demostrando una vez más que el uso de inteligencia artificial, aplicando secuencias lógicas genera mejores soluciones que la búsqueda arbitraria.

La presente investigación emplea en el ajuste de parámetros una técnica que incursiona en los terrenos de la inteligencia artificial, rumbo al diseño de algoritmos capaces de auto gestionarse y auto corregirse.

9. RECOMENDACIONES

Además de explorar mejoras subsecuentes en el funcionamiento del nuevo algoritmo, Se plantea la posibilidad, para trabajos futuros, de desarrollar versiones de ESSA para variable entera y variable permutada. Además de estudiar la posibilidad de convertir al algoritmo de Estrategias de Escape, en una lógica capaz de auto gestionar el ajuste de sus parámetros.

Se plantea además la posibilidad, para trabajos futuros, de realizar una comparación entre los métodos estadísticos de ajuste de parámetros, y la parametrización con inteligencia artificial.

10. BIBLIOGRAFÍA.

Abbass, H. a., 2001. MBO: marriage in honey bees optimization-a Haplometrosis polygynous swarming approach. *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, 1, pp.207–214. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=934391.

Arora, S. y Singh, S., 2013. A conceptual comparison of firefly algorithm, bat algorithm and cuckoo search. *2013 International Conference on Control, Computing, Communication and Materials, ICCCCM 2013*, (Iccccm), pp.1–4.

Baudry, B. et al., 2005. Automatic test case optimization: A bacteriologic algorithm. *IEEE Software*, 22, pp.76–82.

Bidar, M. y Rashidy Kanan, H., 2013. Jumper firefly algorithm. *Iccke 2013*, pp.267–271. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6682839.

Blum, C. y Roli, A., 2003. Metaheuristics in Combinatorial Optimization : Overview and Conceptual Comparison. , 35(3), pp.268–308.

CAMPBELL, D.y STANLEY, J.C., 1996. Experimental and quasi-experimental designs for research.

Chelouah, R. y Siarry, P., 2000. A Continuous Genetic Algorithm Designed for the Global Optimization of Multimodal Functions. , 213, pp.191–213.

Cheng, M.Y. y Prayogo, D., 2014. Symbiotic Organisms Search: A new metaheuristic optimization algorithm. *Computers and Structures*, 139, pp.98–112. Available at: http://dx.doi.org/10.1016/j.compstruc.2014.03.007.

Dorigo, M., 1992. , *learning and naturalalgorithms*. Politecnico di Milano.

Eberhart, J.K., 1995. Particle Swarm Optimization.

Eberhart, R. y Kennedy, J., A New Optimizer Using Particle Swarm Theory. , pp.39–43.

Eberhart, R. y Xi, Y.S., 2001. Particle swarm optimization : Development , applications and resources. *IEE explore*, (February 2001).

Eskandar, H. et al., 2012. Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures*, 110-111, pp.151–166. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0045794912001770.

F C Yang, y P.W., 2007. WATER FLOW-LIKE ALGORITHM FOR OBJECT GROUPING PROBLEMS - Journal of the Chinese Institute of Industrial Engineers - Volume 24, Issue 6.

Ferrer, J., 2010. Metodología De La Investigacion_ TIPOS DE INVESTIGACION Y DISEÑO DE INVESTIGACION.

Figielska, E., 2009. A genetic algorithm and a simulated annealing algorithm combined with column generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources. *Computers & Industrial Engineering Research, 2009,pp.142-151*, p.2009.

Fong, S. et al., 2016. Eidetic Wolf Search Algorithm with a global memory structure. , 254, pp.19–28.

Fukayama, Y. y Yoshida, H.(, 2001. Xplore Abstract - Particle swarm optimization developments, applications and resources. Available at: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=934374&url=http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=934374.

García Sánchez, A., *Técnicas metaheurísticas*, Available at: http://www.iol.etsii.upm.es/arch/metaheuristicas.pdf.

Glover, F., 1986. PATHS FOR INTEGER PROGRAMMING. , 13(5), pp.533–549.

Glover, F., 1989. Tabu Search-Part I. , 1(3), pp.190–206.

Glover, F., 1990. Tabu Search-Part I1. , 2(I), pp.4–32.

Glover, F. y Laguna, M., 1997. *Tabu Search*, Kluwer Academic Publishers Norwell, MA, USA. Available at: file:///D:/desk/art/pages/Tabu Search.htm.

Hajipour, H. et al., 2012. ODMA : A New Metahueristic Optimization Algorithm Based On Open Source Development Model. *2012 12Th International Conference on Intelligent Systems Design and Applications (Isda)*, pp.758–763.

Holland, J.H., 1975a. Adaptation in Natural and Artificial Systems The MIT Press. Available at: https://mitpress.mit.edu/books/adaptation-natural-and-artificial-

systems.

Holland, J.H., 1975b. Genetic Algorthms.

Houck, C.R. y Kay, M.G., 2008. A Genetic Algorithm for Function Optimization : A Matlab Implementation. , (919).

I. Rechenberg, 1965. Cybernetic solution path of an experimental problem, in: RoyalAircraft Establishment Translation. , (1122), p.1965.

Jordehi, A.R., 2015. Enhanced leader PSO (ELPSO): A new PSO variant for solving global optimisation problems. *Applied Soft Computing Journal*, 26, pp.401–417. Available at: http://dx.doi.org/10.1016/j.asoc.2014.10.026.

Kaveh, A. & Khayatazad, M., 2012. A new meta-heuristic method : Ray Optimization. *Computers and Structures*, 112-113, pp.283–294. Available at: http://dx.doi.org/10.1016/j.compstruc.2012.09.003.

Kaveh, A. & Mahdavi, V.R., 2014. Colliding bodies optimization: A novel meta-heuristic method. *Computers & Structures*, 139, pp.18–27. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0045794914000935.

Kirkpatrick, S., M., C.D.G. y Vecchi, M.P., 1983. Optimization by Simulated Annealing. *Science, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680.*, 220(4598), pp.671–680. Available at: http://links.jstor.org/sici?sici=0036-8075(19830513)3:220:4598<671:OBSA>2.0.CO;2-8.

Krus, P. & Andersson, J., 2016. Optimizing Optimization for Design Optimization. , pp.1–10.

Kuo, R.J. & Zulvia, F.E., 2015. The gradient evolution algorithm: A new metaheuristic. *Information Sciences*, 316, pp.246–265. Available at: http://www.sciencedirect.com/science/article/pii/S0020025515002996.

Lence, I.P., 2007. *Técnicas Paralelas Aplicadas a Optimización no Lineal en sistemas de Memoria Distribuida*. Santiago de Compostelas: Universidad de Santiago de Compostelas. Available at: https://books.google.com.co/books?id=cSHQJuonXnUC&pg=PP4&lpg=PP4&dq=Técnicas+Paralelas+Aplicadas+a+Optimización+no+Lineal+en+sistemas+de+Memoria+Distribuida,+Universidad+Santiago+de+Compostelas.&source=bl&ots=_AW6Fd7eLw&sig=ajm0uNjX0Ak1Br4OUwEwF.

Lessmann, S., Caserta, M. y Arango, I.M., 2011. Tuning metaheuristics: A data mining based approach for particle swarm optimization. *Expert Systems with Applications*, 38(10), pp.12826–12838. Available at:

http://dx.doi.org/10.1016/j.eswa.2011.04.075.

Lobo, R.S. & Castro, A.M., 2011. *Cromático, un nuevo método de optimización que se fundamenta a traves de un algoritmo de busqueda basado en la escala cromatica de las notas musicales*. Universidad de Córdoba.

López-ibáñez, M. et al., 2016. The irace package : Iterated racing for automatic algorithm configuration. , 3, pp.43–58.

Maier, H.R. et al., 2014. Environmental Modelling & Software Evolutionary algorithms and other metaheuristics in water resources : Current status , research challenges and future directions * , **. *Environmental Modelling and Software*, 62, pp.271–299. Available at: http://dx.doi.org/10.1016/j.envsoft.2014.09.013.

Martí, R., *Procedimientos Metaheurísticos en Optimización Combinatoria*. Valencia: Universidad de Valencia.

Melián, B. et al., 2003. Metaheurísticas : una visión global *. , 19(19), pp.7–28.

Meng, X.-B. et al., 2015. A novel bat algorithm with habitat selection and Doppler effect in echoes for optimization. *Expert Systems with Applications*, 42(17-18), pp.6350–6364. Available at: http://www.sciencedirect.com/science/article/pii/S0957417415002560.

Merrikh-Bayat, F., 2015. The runner-root algorithm : A metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature. *Applied Soft Computing Journal*, 33, pp.292–303. Available at: http://dx.doi.org/10.1016/j.asoc.2015.04.048.

Monstgomery, D., 2004. Design and Analysis of Experiments Book.pdf. , pp.107–108,649–650.

Moosavian, N. y Kasaee Roodsari, B., 2014. Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. *Swarm and Evolutionary Computation*, 17, pp.14–24. Available at: http://dx.doi.org/10.1016/j.swevo.2014.02.002.

Moscato, P. y Cotta, C., 2004. A Gentle Introduction to Memetic Algorithms. , pp.1–36.

Mousavi, S. et al., 2013. A Hybrid Simulated Annealing Algorithm for Location of Cross-Docking Centers in a Supply Chain.

Niu, J. et al., 2015. Fruit fly optimization algorithm based on differential evolution and its application on gasification process operation optimization. *Knowledge-Based Systems*, 88, pp.253–263. Available at:

http://linkinghub.elsevier.com/retrieve/pii/S0950705115002816.

Pérez, J.A.M., Metaheurísticas : Concepto y Propiedades. Available at: http://www.tebadm.ulpgc.es/almacen/seminarios/MH Las Palmas 2.pdf.

Sadic M. Sait, H., Wiley Iterative Computer Algorithms with Applications in Engineering Solving Combinatorial Optimization Problems - Sadiq M. Available at: http://www.wiley.com/WileyCDA/WileyTitle/productCd-0769501001.html.

Saji, Y., Riffi, M.E. y Ahiod, B., 2014. Discrete bat-inspired algorithm for travelling salesman problem. *2014 Second World Conference on Complex Systems (WCCS)*, pp.28–31. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7060983.

Salimi, H., 2015. Stochastic Fractal Search: A powerful metaheuristic algorithm. *Knowledge-Based Systems*, 75, pp.1–18. Available at: http://www.sciencedirect.com/science/article/pii/S0950705114002822.

Sampieri, R., 2010. *Metodología dela investigación*,

Santana, J.B. et al., 2004. *Metaheurísticas : una revisión actualizada*, Laguna.

Storn, R., 1997. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. , pp.341–359.

Storn, R. y Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, pp.341–359. Available at: http://link.springer.com/article/10.1023/A:1008202821328.

Tang, R., Fong, S. & Deb, S., 2012. Wolf Search Algorithm with Ephemeral Memory. , pp.165–172.

Wang, X. & Zhou, N., 2014. Pattern Search Firefly Algorithm for Solving Systems of Nonlinear Equations. *2014 Seventh International Symposium on Computational Intelligence and Design*, pp.228–231. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7081977.

Yang, X., 2010. A New Metaheuristic Bat-Inspired Algorithm. *Nicso 2010*, pp.65–74.

Yazdani, M. & Jolai, F., 2015. Lion Optimization Algorithm (LOA): A Nature-Inspired Metaheuristic Algorithm. *Journal of Computational Design and Engineering*, pp.1–14. Available at: http://linkinghub.elsevier.com/retrieve/pii/S2288430015000524.

Zanakins, S. H. & Evans, J.R., 1981. Heuristic Optimization: Why, When and How to Use It". Available at:

http://pubsonline.informs.org/doi/abs/10.1287/inte.11.5.84?journalCode=inte.

Zanakis, S.H. & Evans, J.R., 1981. Heuristic " Optimization ": Why , When , and How to Use It WHY , WHEN , AND HOW TO USE IT. *Interfaces*, (August 2015).

11. ANEXOS

11.1. ANEXO 1: FUNCIONES DE PRUEBA, EXPRESIONES MATEMÁTICAS

Tabla 18. Expresiones matemáticas funciones de prueba.

1) Aluffi-Pentini's Problem (AP) (Aluffi-Pentini et al., 1985)
$\min_x f(x) = 0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2$ S.A. $-10 \leq x_1, x_2 \leq 10$. Óptimo: $f(-1.0465, 0) \approx -0.3524.$
2) Bohachevsky 1 Problem (BF1) (Bohachevsky et al. 1986)
$\min_x f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$ S.A. $-50 \leq x_1, x_2 \leq 50$. Óptimo: $f(0,0) = 0.$
3) Bohachevsky 2 Problem (BF2) (Bohachevsky et al., 1986)
$\min_x f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$ S.A. $-50 \leq x_1, x_2 \leq 50$. Óptimo: $x^* = (0,0)$ con $f(x^*) = 0$
4) Camel Back – 6 Six Hump Problem(CB6) (Dixon and Szego , 1978;Michalewicz, 1996)
$\min_x f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$ S.A. $-5 \leq x_1, x_2 \leq 5$. Óptimo: $f(0.89842, -0.712656)$ y $f(-0.089842, 0.712656) = -1.0316.$
5) Camel Back – 3 Three Hump Problem (CB3) (Dixon and Szego, 1975)
$\min_x f(x) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$

S.A.$-5 \leq x_1, x_2 \leq 5$, Óptimo: $f(0,0,0,\ldots,0) = 0$

6) Modified Rosenbrock Problem (MRP) (Price, 1977)

$$\min_{x} f(x) = 100(x_2 - x_1^2)^2 + [6.4(x_2 - 0.5)^2 - x_1 - 0.6]^2$$

S.A.$-5 \leq x_1, x_2 \leq 5$. Óptimo: $f(\ 0.3412\ , 0.1164)\ y\ f(1,1) = 0.$

7) Periodic Problem (PRD) (Price, 1977)

$$\min_{x} f(x) = 1 + \sin^2 x_1 + \sin^2 x_2 - 0.1 \exp\ (-x_1^2 - x_2^2)$$

S.A.$-10 \leq x_1, x_2 \leq 10$, Óptimo: $f(0.0) =\ 0.9.$

8) Schaffer 1 Problem (SF1) (Michalewicz, 1996)

$$\min_{x} f(x) = 0.5 + \frac{\left(\sin\sqrt{x_1^2 + x_2^2}\right)^2 - 0.5}{\left(1 + 0.001(x_1^2 + x_2^2)\right)^2}$$

S.A.$-100 \leq x_1, x_2 \leq 100$. Óptimo: $f(0,0) = 0$

9) Schaffer 2 Problem (SF2) (Michalewicz, 1996)

$$\min_{x} f(x) = (x_1^2 + x_2^2)^{0.25}(\ \sin^2(50(x_1^2 + x_2^2\ )^{0.1}) + 1)$$

S.A.$-100 \leq x_1, x_2 \leq 100$, Óptimo: $f(0,0) = 0$

10) Rastrigin Problem (RG) (Storn and Price, 1997; Torn and Zilinskas, 1989)

$$\min_{x}\ (fx)\ = 10n + \sum_{i=1}^{n} [Xi^2 -\ 10\cos(2\pi Xi)]$$

S.A.$-5.12 \leq Xi \leq 5.12\ , i \in \{1,2,\ldots,n\}$. Óptimo: $f(0,0,\ldots,0) = 0$

11) Ackley's Problem (ACK) (Storn and Price, 1997)

$$\min_{x} f(x) =\ -20\exp\left(-0.02\sqrt{n^{-1}\sum_{i=1}^{n} xi^2}\right) -\ \exp\left(n^{-1}\sum_{i=1}^{n}\cos(2\pi xi)\right) + 20 +\ e$$

S.A.$-30 \leq xi \leq 30, i\ \in\ \{1,2,\ldots,n\}$. Óptimo $f(0,0,0,\ldots,0) = 0.$

12) Exponential Problem (EXP) (Breiman and Cutler, 1993)

$$\min_{x} f(x) = -\exp\left(-0.5\ \sum_{i=1}^{n} x_i^2\right)$$

S.A.$-1 \leq xi \leq 1, i \in \{1,2,\dots,n\}$.El valor óptimo$f(0,0,0,\dots,0) = -1$ se encuentra localizado.
13) Rosenbrock Problem (RB) (Schwefel, 1995; More, et al., 1981)
$\min_x f(x) = \sum_{i=1}^{n-1} [100(x_i + 1 - x_i^2)^2 + (x_i - 1)^2]$ S.A.$-30 \leq xi \leq 30$, i $\in \{1,2,\dots,n\}$. Óptimo: $f(1,1,\dots,1) = 0$
14) Salomon Problem (SAL) (Salomon, 1995)
$\min_x f(x) = 1 - \cos(2\pi\|x\|) + 0.1\|x\|$ Donde $\|x\| = \sqrt{\sum_{i=1}^{d} x_1^2}$ y -100$\leq xi \leq 100$. Óptimo: $f(0,0,0,\dots,0) = 0$
15) Sphere
$\min_x f(x) = \sum_{i=1}^{d} (x_i)^2$ S.A.$-5.12 \leq Xi \leq 5.12, i \in \{1,2,\dots,d\}$. Óptimo: $f(0,0,\dots,0) = 0$
16) Weighted Sphere
$\min_x f(x) = \sum_{i=1}^{d} i(x_i)^2$ S.A.$-100 \leq Xi \leq 100, i \in \{1,2,\dots,d\}$. Óptimo: $f(0,0,\dots,0) = 0$
17) Sum of different power
$\min_x f(x) = \sum_{i=1}^{d} \|x_i\|^{i+1}$ S.A.$-1 \leq Xi \leq 1, i \in \{1,2,\dots,d\}$. Óptimo: $f(0,0,\dots,0) = 0$
18) Step Function
$\min_x f(x) = \sum_{i=1}^{d} \|x_i + 0.5\|^2$ S.A.$-100 \leq Xi \leq 100, i \in \{1,2,\dots,d\}$. Óptimo: $f(-0.5,-0.5,-0.5,\dots,-0.5) = 0$
19) Schwefel 2.2

$\min_{x} f(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j \right)^2$ S.A. $-100 \leq Xi \leq 100$, $i \in \{1,2,\dots,d\}$. Óptimo: $f(0,0,0,\dots,0) = 0$
20) Griewank
$\min_{x} f(x) = \frac{1}{4000} \sum_{i=1}^{d} {x_i}^2 - \prod_{i=1}^{d} \cos(\frac{x_i}{\sqrt{i}}) + 1$ S.A. $-600 \leq Xi \leq 600$, $i \in \{1,2,\dots,d\}$. Óptimo: $f(0,0,0,\dots,0) = 0$

11.2. ANEXO 2. REPRESENTACIÓN GRÁFICA, FUNCIONES DE PRUEBA

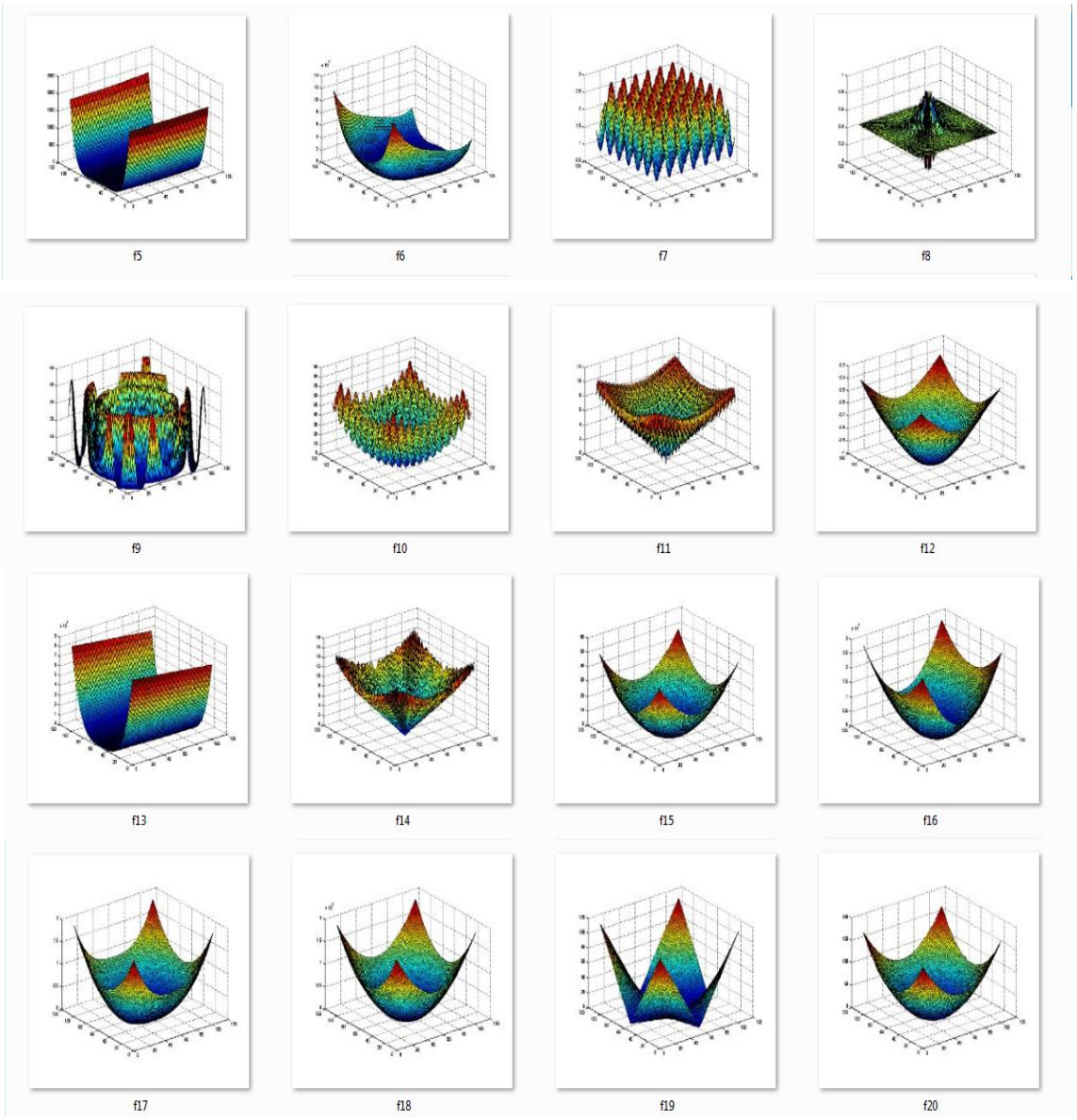

Grafico 2. Ilustración tridimensional de las funciones de prueba

11.3. ANEXO 3. CLASIFICACION TAXONÓMICA DE LOS ALGORITMOS DE PRUEBA

11.3.1. Taxonomía PSO:

El algoritmo de optimización por enjambre de partículas, por su inspiración, se circunscribe en la categoría de "meta heurísticas bio inspiradas", por cuanto imita el comportamiento de una población de organismos vivos; por la naturaleza de sus procedimientos se puede considerar un "algoritmo de búsqueda" por basar su funcionamiento en movimientos de partículas dentro de la región factible. Según el empleo de memoria, PSO actualiza la totalidad de sus registros iteración tras iteración, por lo que puede ser considerada como una meta heurística con memoria a corto plazo; según se realicen o no cambios en la función objetivo, Enjambre de partículas entra en la categoría de "algoritmos de función estática", pues no modifica la función objetivo para realizar la búsqueda de soluciones; Adicionalmente, PSO se constituye en una meta heurística pura pues no se desarrolla a partir de hibridación.

11.3.2. Taxonomía GA:

GA es un algoritmo que el mecanismo de selección natural, por lo que se constituye en una meta heurística bio inspirada; en su funcionamiento hace uso de múltiples estructuras vecinas sin recurrir a modificaciones en la F.O. por lo que entra en la categoría de "algoritmos de función estática". El algoritmo genético no hace un uso extensivo de memoria, por lo que se circunscribe entre los algoritmos con memoria a corto plazo, Por la naturaleza de sus procedimientos, GA es un algoritmo evolutivo.

11.3.3. Taxonomía DE

La mete heurística Evolución diferencial, según la naturaleza de sus procedimientos se circunscribe en la categoría de "algoritmos evolutivos", por cuanto basa su funcionamiento en una población inicial que evoluciona en el tiempo; DE usa memoria a corto plazo y es de función estática, por no realizar cambios en la función objetivo. Evolución diferencial usa múltiples estructuras vecinas y se inspira en el proceso de selección natural por lo que se le considera una meta heurística bio inspirada.

11.3.4. Taxonomía SA

El algoritmo de cristalización simulada o recocido simulado es una meta heurística de inspiración natural no biológica, pues se basa en la analogía de un proceso físico; emplea memoria a corto plazo y múltiples estructuras vecinas, SA es un algoritmo de función estática y se constituye en una meta heurística pura que por la naturaleza de sus procedimientos se circunscribe a la categoría de "algoritmos de búsqueda".

11.3.5. Taxonomía eWSA

El algoritmo EIDETICO, por basar su estructura en la simulación de una interacción natural entre organismos vivos, corresponde a la categoría de "algoritmos bio inspirados"; lo caracteriza un uso extensivo de memoria, por lo que se circunscribe entre las meta heurísticas con memoria a largo plazo. eWSA es además un algoritmo de función estática, pues no modifica la función objetivo para realizar su búsqueda, emplea múltiples estructuras vecinas, y, según la naturaleza de sus procedimientos podemos categorizarlo como un algoritmo de búsqueda.

11.4. ANEXO 4: LÓGICA Y ESTRUCTURA DE LOS ALGORITMOS DE PRUEBA

11.4.1. PSO:

11.4.1.1. Notación:

Sea m_{ij} la j- ésima coordenada (variable de decisión) de la i-ésima partícula, $i \in \{1,2,\dots,n\}, j \in \{1,2,\dots,d\}$ donde n es el número de partículas y d la dimensión (Número de variables de decisión), Sean Li y Ls los límites inferior y superior de las variables de decisión.

Sea $Pbest_{ij}$ la j- ésima coordenada (variable de decisión) de la mejor posición hallada por la i-ésima partícula, Sea v_{ij} la j-ésima componente de la velocidad de la partícula i

Sea $Gbest$ (mejor global) la partícula con mejor evaluación en la función objetivo a lo largo de la ejecución.

11.4.1.2. Parámetros del algoritmo:

Tabla 19. Entradas básicas PSO

Entradas básicas
f: función objetivo
d: dimensión (Número de variables de decisión)
n: Tamaño de la población (manada)
$Stoping\ criteria$: Criterio de parada

Tabla 20. Parámetros de funcionamiento PSO

Parámetros de funcionamiento	
w: Factor de inercia, simula el efecto de la	v: Velocidad de las partículas
c_1: proporción en que cada partícula dará preponderancia a buscar su mejor posición histórica	c_2: Proporción en se dará preponderancia a buscar la posición del mejor global.
$c_1 + c_2 = 1$	

11.4.1.3. Operadores:

A continuación se listan los operadores empleados en Particle Swarm Optimization (PSO):

Tabla 21. Operadores PSO

Entenderemos V(d+1) como el valor en F.O. de un vector V cualquiera		
OPERADOR	**DEFINICIÓN**	**ESTRUCTURA**

Actualizar parámetros	Actualizar w, c_1 y c_2, se recomienda que w y c_1 disminuyan al tiempo que c_2 aumenta para favorecer la exploración en las fases iniciales de la ejecución y para hacer una mejor explotación en las iteraciones finales	`c1=(-1/tiempo)*t+1;`
		`c2=(1/tiempo)*t;`
		`w=(-t/tiempo)+1;`

Mover Enjambre	La posición de cada partícula se actualiza, según la ecuación: m(x,y)=m(x,y)+w*v(x,y)+c1*(pbest(x,y)-m(x,y))*rand+c2*(gbest(y)-m(x,y))*rand;
	Que modela el comportamiento de vuelo en formación de las aves migratorias.
		`for x=1:n`
		`  for y=1:d`
		`  m(x,y)=m(x,y)+v(x,y);`
		`v(x,y)=w*v(x,y)+c1*(pbest(x,y)-m(x,y))*rand+c2*(gbest(y)-m(x,y))*rand;`
		`  end`
		`end`

Actualizar Pbest	Si algún individuo encuentra una posición que mejore su Pbest, este se reemplaza por el nuevo valor; esto para cada partícula	`for x=1:n`
		`if m(x,d+1)<pbest(x,d+1);`
		`  pbest(x,:)=m(x,:);`
		`end`
		`end`

Relevo	Emulando el comportamiento de las parvadas, los individuos toman turnos para ocupar el frente de la formación, en esta analogía, el individuo en frente de formación aquel con mejor F.O.	`if mejor_Pbest(d+1)`
		`<gbest(d+1)`
		`  gbest=mejor_Pbest;`
		`end`

11.4.1.4. Secuencia

Tabla 22. Estructura de código PSO

Entenderemos V(d+1) como la evaluación en F.O.

INICIO

- Inicializar parámetros: f, d, n, li, ls, c1, c2, w, stoping criteria.

Crear población inicial:

- m=li*ones(n,d+1)+(ls-li)*rand(n,d+1);

Crear matriz de velocidades:

- v=rand(n,d+1);
- Evaluar(m)
- Pbest=m;
- Gbest=Individuo_con_mejor_evaluacion_en_Pbest

  Mientras no se cumpla Stoping criteria
  - Actualizar_parámetros
  - Mover_enjambre
  - Evaluar(m)
  - Actualizar_Pbest
  - Relevo

  Finmientras

- Mostrar Gbest
- **FIN PROCESO**

11.4.2. GA

11.4.2.1. Notación:

Sea m_{ij} la matriz de población, que contiene el j-ésimo gen del i-ésimo individuo, con $i \in \{1,2,\dots,n\}$ y $j \in \{1,2,\dots,d\}$ donde n es el número de individuos en la población y d es la cantidad de genes (dimensión o número de variables de decisión).

Sean Li y Ls los respectivos limites inferior y superior de las variables de decisión.

Sean mp la matriz de padres y mh la matriz de hijos, en la matriz de padres se guarda la información genética de aquellos individuos que son seleccionados para transmitir sus genes a la siguiente generación, y la matriz de hijos los nuevos individuos creados a partir de los genes de los padres.

Entenderemos por fitness el valor en la función objetivo de un cierto individuo.

Sea $mejv$ el individuo con mejor fitness hallado en la ejecución.

11.4.2.2. Parámetros del algoritmo:

Tabla 23. Entradas básicas GA

Entradas básicas
f: función objetivo
d: dimensión (Número de variables de decisión)
n: Tamaño de la población (manada)
$Stoping\ criteria$: Criterio de parada

Tabla 24. Parámetros de funcionamiento GA

Parámetros de funcionamiento	
$pm \in \mathbb{R}[0,1]$: Probabilidad de mutación	Tor: Número de individuos por torneo, el operador de selección escogido es el de torneos, Torneos, donde solo los individuos mejor adaptados (con mejor fitness) de entre una muestra aleatoria transmite sus genes

11.4.2.3. Operadores

Tabla 25. Operadores GA

Entenderemos V(d+1) como el valor en F.O. de un vector V cualquiera		
OPERADOR	**DEFINICIÓN**	**ESTRUCTURA**

Selección	Se escogen los individuos en la población que integrarán la matriz de padres y transmitirán sus genes, la probabilidad de un individuo de ser seleccionado es proporcional a su fitness. En este trabajo se empleó el método de selección por torneos	`for i=1:n`
		`r=randi(n,tor,1);`
		`se llena la lista del torneo`
		`for x=1:tor`
		`rr(x,:)=m(r(x),:);`
		`end`
		`rr=evaluarfo(f,rr);`
		`se escoje al mejor individuo`
		`[~,pos]=min(rr(:,d+1));`
		`mp(i,:)=rr(pos,:);`
		`end`

Cruzamiento	Se combinan los genes de los padres para formar nuevos individuos, en el presente trabajo se empleó un operador de cruzamiento por punto de	`for i=1:2:n` *(para i=1 hasta n de 2 en 2)*
		`r=randi(d-1);` *Aleatorio entero entre 1 y d-1 (punto de corte)*
		`mh(i,1:r)=mp(i,1:r);`
		`mh(i,r+1:d)=mp(i+1,r+1:d);`

	corte variable. De cada pareja de padres nace una pareja de hijos.	`mh(i+1,1:r)=mp(i+1,1:r);`
		`mh(i+1,r+1:d)=mp(i,r+1:d);`
		`end`

Mutación	Cada individuo posee una probabilidad de mutación, la mutación consiste en cambiar uno o más genes aleatorios por valores escogidos según una distribución uniforme de probabilidad. Esto simula como la naturaleza prueba nuevas configuraciones genéticas no heredadas.	`for i=1:n`
		`if (rand<pm)`
		`r=randi(d);`
		`rr=randperm(d,r);`
		`for j=1:r`
		`mh(i,rr(j))=li+(ls-li)*rand;`
		`end`
		`end`
		`end`

Relevo generacional	La matriz de hijos pasará a ser la nueva población, para competir por convertirse en padres	`m=mh;`

Actualización	Si se encuentra un individuo con mejor fitness que la mejor solución actual, dicho individuo pasará a reemplazar a $mejv$	`[~,pos]=min(m(:,d+1));`
		`mejor=m(pos,:);`
		`if mejorv(d+1)<mejv(d+1)`
		`mejv=mejorv;`
		`end`

11.4.2.4. Secuencia

Tabla 26. Estructura de código GA

Entenderemos V(d+1) como la evaluación en F.O.

INICIO

- Inicializar parámetros: f, d, n, li, ls, c1, c2, w, stoping criteria.

Crear población inicial:

- m=li*ones(n,d+1)+(ls-li)*rand(n,d+1);
- Evaluar(m)
- v=rand(n,d+1);

Inicializar mp y mh

- mp=inf(n,d+1);
- mh=mp;
- Evaluar(m)
- mejv=mejor individuo en m

  Mientras no se cumpla Stoping criteria
  - selección
  - cruzamiento
  - mutación
  - Evaluar(mh)
  - Relevo_generacional
  - Actualizar mejv

  Finmientras

- Mostrar mejv
- **FIN PROCESO**

11.4.3. DE

11.4.3.1. Notación

Sea X_{ij} el valor de la j-ésima variable de decisión correspondiente al i-ésimo individuo de la población, donde $i \in \{1,2,\dots,n\}$ y $j \in \{1,2,\dots,d\}$, siendo n el número de individuos en la población y d la dimensión (número de variables de decisión).

Sea $X2_{ij}$ un vector temporal que alberga a los individuos luego de ser mutados.

Sea $Trial$ un vector transitorio que albergará los resultados de la mutación.

Sea $Mejv$ la mejor solución hallada

Sea F.O. la función objetivo, entendida como el Fitness de un individuo

11.4.3.2. Parámetros del algoritmo

Tabla 27. Entradas básicas DE

Entradas básicas
f: función objetivo
d: dimensión (Número de variables de decisión)
n: Tamaño de la población (manada)
$Stoping\ criteria$: Criterio de parada

Tabla 28. Parámetros de funcionamiento DE

Parámetros de funcionamiento	
$a, b, c \in \mathbb{N}[1, n]$: Números aleatorios enteros y positivos distintos entre sí.	$CR \in \mathbb{R}[0,1]$: Constante de cruzamiento, determinada por el usuario.
$F \in \mathbb{R}[0,2]$: Factor real constante	c_2: Proporción en se dará preponderancia a buscar la posición del mejor global.

11.4.3.3. Operadores

Tabla 29. Operadores DE

Entenderemos V(d+1) como el valor en F.O. de un vector V cualquiera		
OPERADOR	**DESCRIPCIÓN**	**ESTRUCTURA**

Selección	Elegir al azar tres individuos distintos de la población	`a=randi(n);`
		`b=randi(n);`
		`c=randi(n);`
		`while b==a`
		`b=randi(n);`
		`end`
		`while or(c==a,c==b)`
		`c=randi(n);`
		`end`

Mutación	Asigna un probabilidad a un determinado gen en posición *j*, de sufrir una mutación, el nuevo valor se calcularía a partir de los individuos en las posiciones a y b, y guardándose en $trial$	`if or (rand<CR,k==d)`
		`trial(j)=x(i,j)+EFE*(x(a,j)-x(b,j));`
		`else`
		`trial(j)=x(i,j);`
		`j=mod(j+1,d)+1;`
		`end`

Cruzamiento	Para cada individuo, si su versión mutada es mejor que la versión original del vector, el vector mutado pasa a reemplazar al individuo inicial.	`if trial(d+1)<x(i,d+1)`
		`x2(i,:)=trial;`
		`else`
		`x2(i,:)=x (i,:);`
		`end`
		`x=x2;`

Actualización	Si el mejor individuo de la población ($mejorv$) logra una	`if mejorv(d+1)<mejv(d+1)`
		`mejv=mejorv;`
		`end`

	mejor evaluación en F.O. que $mejv$, entonces dicho individuo pasa a reemplazar a $mejv$.	

11.4.3.4. Secuencia

Tabla 30. Estructura de código DE

Entenderemos v(d+1) como el valor en F.O. de un vector V cualquiera

INICIO

- Inicializar parámetros: f, d, n, li, ls, a, b, c, CR, F, stoping criteria.

Crear población inicial:

- m=li*ones(n,d+1)+(ls-li)*rand(n,d+1);
- Evaluar(m)
- Calcular_mejv (individuo con mejor evaluación en F.O.)

  Mientras no se cumpla Stoping criteria
  - selección
  - j=aleatorio_entero_entre_1_y_d

    para k=1 hasta d
    - mutación

    finpara
  - Evaluar(trial)
  - Cruzamiento
  - Actualización

  Finmientras

- Mostrar mejv
- **FIN PROCESO**

11.4.4. SA

11.4.4.1. Notación

Sea X_{ij} la j-ésima variable de decisión de la i-ésima partícula, donde $i \in \{1,2,\dots,n\}$ y $j \in \{1,2,\dots,d\}$ con n el número partículas y d la dimensión (Número de variables de decisión del problema).

Sean Li y Ls los límites inferior y superior de las variables de decisión

Sea $mejv$ la mejor configuración hallada.

Sea F.O. la función objetivo entendida como la energía de la configuración.

11.4.4.2. Parámetros del algoritmo

Tabla 31. Entradas básicas SA

Entradas básicas
f: función objetivo
d: dimensión (Número de variables de decisión)
n: Tamaño de la población (manada)
$Stoping\ criteria$: Criterio de parada

Tabla 32. Parámetros de funcionamiento SA

Parámetros de funcionamiento	
$t0$: Temperatura inicial, por lo general un valor alto	$a \approx 1, a \in \mathbb{R}[0,1]$: Factor de enfriamiento, mientras más cercano a 1, más lento es el descenso de la temperatura.

11.4.4.3. Operadores

Tabla 33. Operadores SA

Entenderemos V(d+1) como el valor en F.O. de un vector V cualquiera		
OPERADOR	**DEFINCIÓN**	**ESTRUCTURA**

Generar vecino	Se crea un nuevo vector en la vecindad de la partícula anterior	`for i=1:n`
		`ran=randi(d);`
		`xx(i,ran)=li+(ls-li)*rand;`
		`end`

Evaluar delta de temperatura	Si la nueva partícula posee una menor energía, este nuevo vector reemplaza a la partícula anterior, de lo contrario se calcula el delta de temperatura para la partícula y si este queda por debajo de la curva de la distribución continua de Boltzman, el valor es admitido, de lo contrario la partícula inicial no se altera	`r=rand;`
		`delta=abs(xx(i,d+1)-`
		`x(i,d+1));`
		`if r<exp(-(delta/temp))`
		`  x(i,:)=xx(i,:);`
		`end`

Actualización:	Si la partícula con menor energía mejora la solución global, esta pasa a reemplazar a *mejv*.	`[~,posicion]=min(x(:,d+1));`
		`mejorv=x(posicion,:);`
		`if mejorv(d+1)<mejv(d+1)`
		`  mejv=mejorv;`
		`end`

Discutir temperatura	Utilizando una serie geométrica disminuimos paulatinamente la temperatura, lo que hace que la probabilidad de aceptar soluciones que empeoren las configuraciones actuales baje, ello por efectos de las características de la distribución de Boltzman	`temp=temp*a;`

11.4.4.4. Secuencia

Tabla 34. Estructura de código SA

Entenderemos V(d+1) como la evaluación en F.O. de un vector v cualquiera, la notación V(i,:) se entenderá como [v1,v2,…,vd]

INICIO

- Inicializar parámetros: f, d, n, li, ls, t0, a, stoping criteria.

Crear población inicial:

- m=li*ones(n,d+1)+(ls-li)*rand(n,d+1);
- Evaluar(m)
- Calcular_mejv (individuo con mejor evaluación en F.O.)

Mientras no se cumpla Stoping criteria

para i=1 hasta n

- Y=Generar_vecino_de_x(i,:)
  - Evaluar(Y)

Si Y(d+1)<x(d+1)

- X(i,:)=Y

Sino

- Evaluar_delta_de_temperatura

Finsi

Finpara

- Evaluar(X)
- Actualizar
- Disminuir temperatura

Finmientras

- Mostrar mejv
- **FIN PROCESO**

11.4.5. eWSA

11.4.5.1. Notación

Sea X_{ij} la j-ésima variable de decisión del i-ésimo individuo (Lobo), donde $i \in \{1,2,\dots,n\}$ y $j \in \{1,2,\dots,d\}$ con n el número de lobos en la manada y d la dimensión (Número de variables de decisión del problema).

Sea Xi_{ij} un vector transitorio de $1Xd$ que guarda valores temporales de X_i Para un i determinado

Sean Li y Ls los límites inferior y superior de las variables de decisión

Sea $best\_loc$ la mejor configuración hallada.

11.4.5.2. Parámetros del algoritmo

Tabla 35. Entradas básicas eWSA

Entradas básicas
f: función objetivo
d: dimensión (Número de variables de decisión)
n: Tamaño de la población (manada)
$Stoping\ criteria$: Criterio de parada

Tabla 36. Parámetros de funcionamiento eWSA

Parámetros de funcionamiento	
$p \in \mathbb{R}[0,1]$: probabilidad de que un lobo no salga de la localidad	$a \in \mathbb{R}[0,1]$: factor de velocidad de los lobos
$s \in \mathbb{R}[0,1]$: Distancia para escapar de la localidad	$v \in \mathbb{R}[0,1]$: Radio del rango visual
$u \in \mathbb{N}$: $Tamaño\ de\ la\ lista\ tab$ú	$fdeholgura \in \mathbb{R}[0,1]$: Determina si una solución v es tachada como tabú, si un vector Y cualquiera cumple que $\lvert Y_j - TL_{ij}\rvert < fdeholguras\ \forall j$ y para algún i, no será tachado. TL_{ij} es la j-ésima componente del i-ésimo individuo en la lista tabú, denotado en el código como $tabulist_{ij}$

11.4.5.3. Operadores

Tabla 37. Operadores eWSA

Entenderemos V(d+1) como el valor en F.O. de un vector V cualquiera		
OPERADOR	**DEFINICIÓN**	**ESTRUCTURA**

Prey New Food initiatively	Nueva presa Inicial, genera una posición vecina a la que se mueve el lobo en busca de una nueva presa, esta nueva posición no puede estar entre los valores tachados como tabú	radio=a*v;
		yi=xi;
		Ran=randi(d);
		$Yi\,(ran) = \text{radio} * (\text{li} + (\text{ls} - \text{li}) * \text{rand});$
		while $Yi \in Tabulist$
		$Yi = Xi;$
		Ran=randi(d);
		$Yi\,(Ran) = \text{radio} * (\text{li} + (\text{ls} - \text{li}) * \text{rand});$
		end

prey_new_food_pasively	Modo pasivo: se genera valor aleatorio en el intervalo [v,ls/2], y se reemplaza en una posición aleatoria j del individuo, este nuevo vector no puede estar entre los valores tachados como tabú.	escape=v+(ls*0.5-v)*rand;
		vi=xi+a*s*escape;
		Ran= randi(d);
		yi(ran)=vi(ran);
		while yi $\in Tabulist$
		escape=v+(ls*0.5-v)*rand;
		vi=xi+a*s*escape;
		yi(ran)=vi(ran);
		end

escape_region	Se genera un movimiento amplio para evitar estancamiento en óptimos locales, la nueva posición no debe encontrarse entre los valores tachados como tabú.	yi=xi+a*s*rand(1,d+1)*((-1)^randi(2));
		while yi $\in Tabulist$
		yi=xi+a*s*rand(1,dd)*((-1)^randi(2));
		end

update_the_best_loc	Si se encuentra una mejor solución que la actual, el nuevo valor pasa a reemplazar a $best\_loc$	[~,pos]=min(x(:,d+1)); Mejorv=x(pos,:); if mejorv(d+1)<best_loc(d+1) best_loc=mejorv; end

update_tabulist	Todos los valores explorados en la iteración actual, con excepción del mejor pasan a formar parte de la $tabulist$ o lista tabú. Se reemplazan los $n-1$ valores más antiguos por los $n-1$ nuevos valores	Mejorv=individuo con mejor evaluación en F.O. de la población actual X
		Y=x;
		[~,pos]=min(Y(:,d+1));
		Mejorv=Y(pos,:);
		auxiliar=tabulist;
		Reemplazar x(pos,:) con otro individuo de la población
		[~,pos]=min(Y(:,d+1));
		algo=randi(n-2)+1;
		while abs(algo-pos)<=1
		algo=randi(n);

		end
		Y(pos,:)=Y(algo);
		Reemplazar valores antiguos en lista tabú
		aux=tabulist;
		for ii=n+1:u
		aux(ii,:)=tabulist(ii-1,:);
		end
		aux(1:n,:)=x;
		tabu=aux;
		end

Distancia	Calcula la distancia entre dos lobos	delta=x1-x2;
		delta1=delta.^2;
		sumar=sum(delta1);
		distancia=sumar^0.5;

11.4.5.4. Secuencia

Tabla 38. Estructura de código eWSA

Entenderemos V(d+1) como la evaluación en F.O. de un vector v cualquiera, la notación V(i,:) se entenderá como [v1,v2,…,vd]

INICIO

- Inicializar parámetros: f, d, n, li, ls, p, a, v, s, u, fdeholguras, stoping criteria.

Crear población inicial:

- m=li*ones(n,d+1)+(ls-li)*rand(n,d+1);
- Evaluar(m)
- Tabulist=inf(u,d+1);
- Calcular_mejv (individuo con mejor evaluación en F.O.)

Mientras no se cumpla Stoping criteria

- Para i=1 hasta n
  - X(i,:)=prey_new_food_initiatively
  - Evaluar(x(i,:))
  - Loc=x(i,:)

  Para j=1 hasta n

  Si distancia(x(i,:),x(j,:))<v y x(j,d+1)<loc(d+1)
    - Loc=x(j,:)

  fisi

  finpara

  si (no (x(i,:)==loc)

  x(i,:)=loc;

  sino

  xi=x(i,:);

  x(i,:)=prey_new_food_pasively

  finsi

  if rand>p
    - xi=x(i,:);
    - x(i,:)=escape_region

  finsi
- x=evaluarfo(f,x);
- best_loc=update_the_best_loc
- tabulist=update_tabulist

  Finpara

Finmientras

- Mostrar mejv
- **FIN PROCESO**

11.5. ANEXO 5: NÚMERO DE CITAS CORRESPONDIENTES A CADA ALGORITMO REVISADO.

Tabla 39. Cantidad de citas correspondiente a cada algoritmo revisado

Nombre	Name	Abreviatura	Citas
Algoritmo genético	Genetic algorithm	GA	50143
Recocido simulado	Simulated annealing	SA	37440
Evolución diferencial	Diferential evolution	DE	11624
Optimización de Enjambre de partículas	Particle swarm optimization	PSO	9902
Colonia de hormigas	Ant colony	ACO	7689
Búsqueda tabú	Tabu search	TS	6823
Búsqueda de cuco	Cuckoo search	CS	1667
algoritmo de búsqueda gravitatoria	gravitational search algorithm	GSA	1593
Rankin estocástico	stochastic rankin	SR	1347
optimización basada biogeografía	biogeography based optimization	BBO	1274
algoritmo murciélago	Bat algorithm	BAT	974
asignaciones Homogéneas	Homomorphus mappings	HM	775
recocido simulado adaptado	adapted simulated annealing	ASA	648
colonia artificial de abejas	artificial bee colony	ABC	621
inspirado en la colonización de malezas	inspired from weed colonization	IWO	548
sencilla estrategia de evolución multi miembro	simple multimembered evolution strategy to solve	SMES	508
luciérnagas	Fire fly	FFA	326
sesgos de búsqueda de optimización evolución restringida	search biases in constrainedevolutionary optimization	ISR	316
evolución diferencial cultivada	cultured differential evolution	CULDE	198

algoritmo segregacional de manejo adaptativo limitado	adaptive segregatiational constrained handling algorithm	ASCHEA	190
evolución diferencial con la selección dinámica estocástico	Diferential evolution with dynamic stochastic selection	DEDS	165
función de penalización auto adaptativa	self adaptaive penalty function	SAPF	147
ciclo del agua	Water cycle	WCA	104
búsqueda armonica	harmony search	HS	103
Búsqueda de caza	hunting search	HUS	94
algoritmo evolutivo híbrido adaptativo restringido	hybrid evolutionary algorithm and adaptative constraint handling	HEAA	90
la mejora de la optimización de mosca de la fruta	improved fruit fly optimization	IFFO	44
Búsqueda de lobo	Wolf search	WSA	41
PSO líder mejorada	Enhanced leader PSO	ELPSO	40
algoritmo genético mejorado	improved Genetic algorithm	IGA	30
Búsqueda Estocástica fractal	Stochastic fractal search	SFS	29
optimización por onda de agua	water wave optimization	WWO	28
algoritmo león	Lion optimization	LOA	13
Recocido estrategia de Nelder y Mead	Annealed Nelder and Mead strategy	ANM	12
luciérnaga con saltos	Jumper firefly	JFF	11
mosca de la fruta	Fruit fly	DFOA	8
gradiente evolución	Gradient evolution	GE	6
Desarrollos modelo de código abierto	Open source developement model	ODMA	2

11.6.ANEXO 6: PROMEDIO CALIDAD DE RESPUESTA, INSTANCIAS VS ALGORITMO.

Tabla 40. Resultados experimento de comparación, promedio muestral Función vs Algoritmo, discriminado por dimensiones

F.O. Promedio para instancias 2D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
1	-0,302	**-0,352**	-0,302	**-0,352**	-0,352	-0,352	-0,352
2	0,000	**0,000**	0,000	**0,000**	0,000	0,000	0,281
3	**0,000**	**0,000**	0,000	**0,000**	0,000	0,000	0,095
4	-1,031	**-1,032**	-1,029	**-1,032**	-1,032	-1,032	-1,031
5	**0,000**	0,000	0,000	0,000	0,000	0,000	0,000
6	0,011	**0,000**	0,014	0,000	0,000	0,000	0,004
7	0,900	**0,900**	0,900	0,989	0,900	0,900	0,901
8	**0,000**	**0,000**	0,002	0,000	0,000	0,007	0,010
9	0,000	**0,000**	0,000	0,000	0,000	0,000	0,000
F.O. Promedio para instancias 5D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
10	4,2E-05	2,0E-13	9,5E-04	**0,0E+00**	6,9E-04	2,2E-01	1,1E+01
11	2,2E-05	1,1E-09	7,2E-03	**1,8E-15**	4,9E-03	2,6E-01	1,6E+00
12	-1,0E+00	-1,0E+00	-1,0E+00	**-1,0E+00**	-1,0E+00	-1,0E+00	-1,0E+00
13	4,0E+00	1,5E+03	3,5E+00	2,1E-01	3,3E-01	**1,8E-01**	4,4E+03
14	**5,9E-03**	1,0E-01	9,5E-02	7,7E-02	1,0E-01	2,3E-01	1,4E+00
15	**0,0E+00**	8,5E-07	1,3E-05	1,1E-52	7,9E-06	1,3E-03	3,7E-01
16	**6,5E-96**	5,5E-03	4,3E-03	7,2E-50	7,0E-03	3,2E-03	3,7E+02
17	**0,0E+00**	2,2E-10	1,2E-08	1,3E-112	9,9E-11	3,9E-06	2,4E-04
18	**0,0E+00**	**0,0E+00**	**0,0E+00**	**0,0E+00**	**0,0E+00**	**0,0E+00**	1,2E+02
19	3,5E-03	8,0E-04	5,0E-03	**1,5E-27**	7,8E-03	6,7E-02	2,3E+00
20	**0,0E+00**	2,1E-02	1,4E-02	9,3E-13	1,9E-02	5,8E-02	2,5E+00
F.O. Promedio para instancias 10D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
10	0,000	4,478	0,064	**0,000**	0,031	4,048	59,668
11	0,001	0,227	0,055	**0,000**	0,036	0,435	3,613
12	-1,000	-1,000	-1,000	**-1,000**	-1,000	-0,998	-0,862
13	8,979	962,812	50,654	10,862	**7,877**	25,066	476794,606
14	**0,008**	0,175	0,129	0,100	0,301	0,450	5,745

15	**0,000**	0,021	0,000	0,000	0,000	0,015	7,499
16	0,000	118,679	0,519	**0,000**	0,251	0,064	11574,659
17	**0,000**	0,000	0,000	0,000	0,000	0,000	0,005
18	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	2637,750
19	0,008	0,345	0,038	**0,000**	0,045	0,274	12,277
20	**0,000**	0,178	0,098	0,006	0,179	0,216	22,720
F.O. Promedio para instancias 20D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
10	**0,001**	19,443	1,978	0,498	0,483	24,039	185,594
11	**0,001**	0,703	0,147	0,205	0,278	0,760	4,613
12	-1,000	-0,995	-1,000	**-1,000**	-1,000	-0,987	-0,441
13	**19,011**	7213,332	155,502	46,442	92,154	487,320	24070063,360
14	**0,048**	0,450	0,519	0,225	0,780	1,078	13,489
15	**0,000**	0,129	0,017	0,000	0,004	0,074	39,217
16	0,018	1542,928	63,635	**0,000**	10,769	1,343	141447,617
17	**0,000**	0,000	0,000	0,000	0,000	0,000	0,028
18	**0,000**	4,500	5,000	**0,000**	1,750	3,000	16646,000
19	0,026	5,298	0,246	**0,000**	0,280	1,027	50,192
20	**0,000**	2,175	0,295	0,003	0,943	0,026	144,274
F.O. Promedio para instancias 50D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
10	**0,005**	197,014	16,582	62,660	20,473	213,729	626,162
11	**0,003**	1,284	0,386	1,008	0,696	2,040	5,922
12	**-1,000**	-0,879	-0,982	-1,000	-0,999	-0,790	-0,021
13	**49,097**	33729,121	19902,914	12201,774	2083,594	4978,088	223767520
14	**0,070**	2,550	1,937	1,902	3,381	3,805	28,830
15	**0,000**	1,944	0,561	0,066	0,215	0,849	215,325
16	**0,222**	26927,074	5398,186	2983,160	1472,125	1549,654	1718990,009
17	**0,000**	0,000	0,000	0,000	0,000	0,002	0,085
18	**0,000**	630,250	367,000	121,500	60,500	235,250	81316,000
19	0,126	28,951	3,321	**0,104**	2,846	7,300	3,676E+12
20	**0,000**	17,961	3,527	0,776	1,897	2,149	716,113

11.7.ANEXO 7: DESVIACIÓN ESTÁNDAR CALIDAD DE RESPUESTA (INSTANCIA VS ALGORITMO)

Tabla 41. Resultados experimento de comparación, Desviación estándar, Función vs algoritmo, discriminado por dimensión.

Desviación estándar F.O. para instancias 2D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
1	0,100	**0,000**	0,100	**0,000**	0,000	0,000	0,000
2	0,000	**0,000**	0,000	**0,000**	0,000	0,000	0,237
3	**0,000**	**0,000**	0,000	**0,000**	0,000	0,000	0,097
4	0,001	**0,000**	0,003	0,000	0,000	0,000	0,000
5	**0,000**	0,000	0,000	0,000	0,000	0,000	0,000
6	0,016	**0,000**	0,022	0,000	0,000	0,000	0,004
7	0,000	**0,000**	0,000	0,022	0,000	0,000	0,001
8	**0,000**	**0,000**	0,005	0,000	0,000	0,005	0,000
9	0,000	**0,000**	0,000	0,000	0,000	0,000	0,000
Desviación estándar F.O. para instancias 5D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
10	4,6E-05	3,1E-13	1,9E-03	**0,0E+00**	2,0E-04	3,0E-02	2,3E+00
11	2,9E-05	2,0E-09	1,4E-02	**1,8E-15**	3,4E-03	1,6E-01	3,7E-01
12	1,2E-08	**0,0E+00**	6,9E-07	**0,0E+00**	4,0E-08	4,5E-05	1,2E-03
13	**2,7E-02**	3,0E+03	1,0E+00	1,7E-01	2,1E-01	1,3E-01	2,4E+03
14	5,0E-03	**0,0E+00**	1,7E-01	4,6E-02	5,2E-04	5,0E-02	4,9E-01
15	**0,0E+00**	1,4E-06	2,6E-05	2,1E-52	1,4E-06	8,7E-04	2,1E-01
16	1,3E-95	6,9E-03	8,7E-03	1,4E-49	1,8E-03	1,6E-03	1,1E+02
17	**0,0E+00**	1,3E-10	2,4E-08	1,6E-112	6,1E-11	2,6E-06	1,3E-04
18	**0,0E+00**	**0,0E+00**	**0,0E+00**	**0,0E+00**	**0,0E+00**	**0,0E+00**	3,9E+01
19	1,5E-03	1,1E-03	7,1E-03	**3,0E-27**	2,5E-03	1,0E-02	3,2E-01
20	**0,0E+00**	4,7E-03	2,7E-02	1,9E-12	4,2E-03	2,5E-02	4,7E-01
Desviación estándar F.O. para instancias 10D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
10	0,000	2,368	0,127	**0,000**	0,011	1,088	1,268
11	0,000	0,099	0,110	**0,000**	0,013	0,059	0,115
12	0,000	0,000	0,000	**0,000**	0,000	0,001	0,030

13	**0,027**	1758,875	83,312	6,727	1,785	38,063	187709,849
14	0,005	0,050	0,247	**0,000**	0,001	0,058	0,811
15	**0,000**	0,012	0,001	0,000	0,000	0,004	0,465
16	0,000	116,097	1,037	**0,000**	0,099	0,011	1320,990
17	**0,000**	0,000	0,000	0,000	0,000	0,000	0,004
18	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	**0,000**	1025,915
19	0,002	0,228	0,057	0,000	0,008	0,037	2,643
20	**0,000**	0,144	0,197	0,004	0,013	0,099	5,700
Desviación estándar F.O. para instancias 20D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
10	**0,000**	8,610	3,954	0,995	0,292	3,636	5,935
11	**0,001**	0,186	0,293	0,088	0,062	0,123	0,377
12	0,000	0,005	0,000	**0,000**	0,000	0,003	0,035
13	**0,011**	10572,993	272,984	30,737	23,567	888,956	1439614,335
14	**0,038**	0,173	0,975	0,050	0,164	0,093	0,690
15	**0,000**	0,058	0,035	0,000	0,001	0,019	2,489
16	0,005	155,289	127,238	**0,000**	4,702	0,526	22360,957
17	**0,000**	0,000	0,000	0,000	0,000	0,000	0,007
18	**0,000**	1,291	10,000	**0,000**	1,500	1,414	2041,921
19	0,014	1,180	0,441	**0,000**	0,055	0,116	2,164
20	**0,000**	0,612	0,591	0,005	0,076	0,007	15,549
Desviación estándar F.O. para instancias 50D							
Función	**ESSA**	**PSO**	**GA**	**DE**	**SA**	**eWSA**	**RANDOM**
10	**0,005**	26,432	33,146	20,019	6,188	18,833	12,285
11	**0,001**	0,058	0,761	0,174	0,084	0,320	0,036
12	**0,000**	0,022	0,036	0,000	0,000	0,051	0,004
13	**0,061**	10412,880	39707,652	22265,803	952,586	3639,130	29931960,490
14	**0,035**	0,238	3,723	0,497	0,330	1,028	0,342
15	**0,000**	0,759	1,122	0,130	0,029	0,163	8,569
16	**0,155**	4072,762	10795,946	4749,222	509,672	1423,087	86538,096
17	**0,000**	0,000	0,000	0,000	0,000	0,001	0,042
18	**0,000**	186,866	734,000	202,237	7,853	72,972	2522,743
19	**0,023**	3,499	6,211	0,122	0,522	0,814	4,59941E+12
20	**0,000**	7,641	7,054	1,109	0,204	0,286	46,350

11.8.ANEXO 8: VERIFICACIÓN DE SUPUESTOS

Tabla 42. Prueba de Normalidad de residuos Shapiro-Wilk

Prueba	*Estadístico*	*Valor-P*
Estadístico W de Shapiro-Wilk	0,589093	0,0

Como se puede evidenciar, Valor P < 0.05, por lo que se rechaza la hipótesis nula de normalidad, de manera que, con un 95% de confianza podemos afirmar que los residuos no se distribuyen aproximadamente normal.

11.9.ANEXO 9: TABLA ITERACIONES/TIEMPO (SEG) VS ALGORITMO

Tabla 43. Iteraciones por segundo para algoritmo

	2D	5D	10D	20D	50D	promedio	desviación
ESSA	53	42	41	34	22	38	11
PSO	1725	960	613	351	167	763	550
GA	48	46	47	40	31	42	7
DE	71	69	67	62	54	65	6
SA	421	403	373	322	236	351	67
eWSA	15	12	10	7	4	10	4

11.10. ANEXO 10: CURVA CARACTERÍSTICA DE PROCESO:

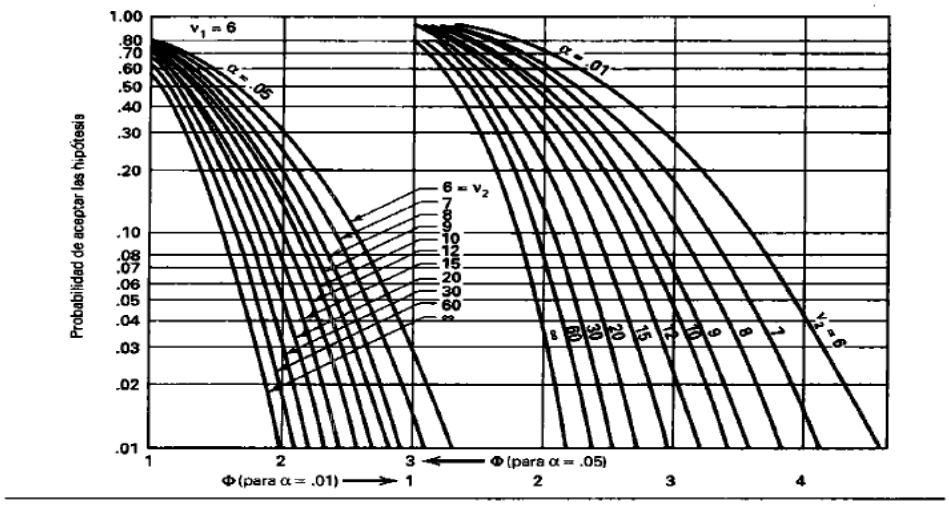

Grafico 3. Curva característica de proceso

11.11. ANEXO 11. CALCULO TAMAÑO DE MUESTRA:

Para el cálculo del tamaño de muestra adecuado se siguió el siguiente procedimiento, expuesto por (Monstgomery 2004):

- Prueba piloto: Se ejecutó cada uno de los 7 algoritmos de comparación para una instancia única (Para tales efectos se usó D5F10, en 10 corridas por algoritmo.
- Calculo de los valores: $\sigma^2, \mu_i, \tau_i \forall i \in \{1,2,\dots,7\}$ según las fórmulas:

$$\mu_i = \frac{1}{10}\sum_{j=1}^{10} X_{ij}\,, \bar{\mu} = \frac{1}{7}\sum_{j=1}^{7} \mu_i, \tau_i = \mu_i - \bar{\mu} \qquad (1)$$

  Donde X_{ij} es el j-ésimo dato obtenido en la corrida del i-ésimo algoritmo, σ^2 es la varianza de X (Para los 70 datos).

- Calculo de los valores $\phi, v_1, v_2, \beta_1, \beta_2$, según el siguiente procedimiento:

$$\phi(n) = \sqrt{\frac{n\sum_{j=1}^{7}\tau^2}{a\sigma^2}}, con\ a = 7\ (Número\ de\ algoritmos) \qquad (2)$$

Tabla 44. Procedimiento de cálculo para el número de réplicas del experimento

n	ϕ	$v_1 = a - 1$	$v_2 = a(n-1)$	β_1	$\beta_2 = 1 - \beta_1$ $(potencia)$
2	$\phi(2)$	$a-1$	$a(2-1)$		
3	$\phi(3)$	$a-1$	$a(3-1)$		
⋮					

El valor β_1 lo conseguimos con las coordenadas (ϕ, v_2) En las curvas características de procesos (Anexo 4), El proceso se detiene cuando $\beta_2 \geq 0.9$ y se toma como tamaño apropiado de muestra el menor n que cumpla dicha condición.

Al aplicar el anterior método al cálculo del tamaño de la muestra se obtuvieron los siguientes datos:

Tabla 45. Cálculo del número de réplicas del experimento de comparación

$\boldsymbol{\sigma^2}$	11,1791209	$\bar{\mu}$	1,35568946

μ_1	μ_3	μ_4	μ_5	μ_6	μ_7
3,9349E-05	4,91518E-13	0,00660018	0	0,00077186	0,259840827
τ_1	τ_3	τ_4	τ_5	τ_6	τ_7
-1,35565011	-1,35568946	-1,34908927	-1,35568946	-1,35491759	-1,09584863

n	ϕ	$v1 = a - 1$	$v2 = a(n-1)$	β_1	$\beta_2 = (1 - \beta_1)$
2	1,35895678	6	7	0,6	0,4
3	1,66437535	6	14	0,23	0,77
4	1,92185511	6	21	0,06	0,94

De esta manera, el número seleccionado de réplicas del experimento fue $n = 4$.

11.12. ANEXO 12: ESPECIFICACIONES DE SISTEMA DEL EQUIPO DE CÓMPUTO DEL EXPERIMENTO

Tabla 46. Especificaciones del sistema del equipo de cómputo en que se ejecutó el experimento de comparación

Nombre del SO	Microsoft Windows 7 Professional
Versión	6.1.7600 compilación 7600
Descripción adicional del SO	No disponible
Fabricante del SO	Microsoft Corporation
Nombre del sistema	ADMIN-PC
Fabricante del sistema	Acer
Modelo del sistema	Aspire E1-431
Tipo de sistema	PC basado en x64
Procesador	Intel(R) Celeron(R) CPU 1005M @ 1.90GHz, 1900 Mhz, 2 procesadores princip...
Versión y fecha de BIOS	Insyde Corp. V2.14, 14/03/2013
Versión de SMBIOS	2.7
Directorio de Windows	C:\Windows
Directorio del sistema	C:\Windows\system32
Dispositivo de arranque	\Device\HarddiskVolume1
Configuración regional	Colombia
Capa de abstracción de hardw...	Versión = "6.1.7600.16385"
Nombre de usuario	admin-PC\admin
Zona horaria	Hora est. Pacífico, Sudamérica
Memoria física instalada (RAM)	4,00 GB
Memoria física total	3,82 GB
Memoria física disponible	2,18 GB
Memoria virtual total	7,64 GB
Memoria virtual disponible	5,45 GB
Espacio de archivo de paginaci...	3,82 GB
Archivo de paginación	C:\pagefile.sys